Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

А. А. Родионов, Д. П. Андреева

УПРАВЛЕНИЕ ВЕБ-ПРОЕКТАМИ

ПОСОБИЕ
по одноименной дисциплине
для слушателей специальности 1-40 01 74
«Web-дизайн и компьютерная графика»
заочной формы обучения

Гомель 2012

УДК 004.738.52(075.8)
ББК 32.973.202я73
Р60

Рекомендовано кафедрой «Информатика» ГГТУ им. П. О. Сухого
(*протокол № 10 от 29.03.2012 г.*)

Рецензенты: начальник центра информационных технологий ГГТУ им. П. О. Сухого
Г. Л. Ильющенко

Родионов, А. А.

Р60 Управление Веб-проектами : пособие по одноим. дисциплине для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заоч. формы обучения / А. А. Родионов, Д. П. Андреева. – Гомель : ГГТУ им. П. О. Сухого, 2012. – 113 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://alis.gstu.by/StartEK. – Загл. с титул. экрана.

Цель данного пособия – овладение современными методологиями и программными средствами проектирования и управления ресурсами при разработке веб-проектов.

Для слушателей ИПК и ПК.

УДК 004.738.52(075.8)
ББК 32.973.202я73

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Веб-разработки являются перспективным направлением деятельности в современной информационной сфере. Стремительный рост числа пользователей интернета и открытие большого числа сайтов привело к тому, что создание и поддержка веб-сайтов стало востребованным направлением IT-технологий. Вместе с тем, в данный момент создание сайтов происходит по далеким от совершенства методикам. В частности, почти полностью отсутствует менеджмент веб-проектов.

Целью методического пособия является повышение профессионального уровня будущих специалистов по веб-дизайну по менеджменту веб-проектов. Задачей методического пособия является формирование у слушателей современных представлений о разработке веб-проектов, привлечении кадров, распределении времени, работой с заказчиком, знаний по решению проблем эффективного использования современных инструментов разработки веб-сайтов на основе мирового опыта веб-дизайна, верстки и веб-программирования.

1. МЕНЕДЖМЕНТ В РАЗРАБОТКЕ ПРОГРАММНЫХ ИЗДЕЛИЙ

В общем виде проект (англ. project) — это «что-либо», что задумывается или планируется, например, программное изделие .

С точки зрения системного подхода, проект может рассматриваться как процесс перехода из исходного состояния в конечное — результат при участии ряда ограничений и механизмов.

В «Кодексе знаний об управлении проектами» проект – некоторая задача с определенными исходными данными и требуемыми результатами, обуславливающими способ ее решения. Проект включает в себя замысел, средства его реализации и получаемые в процессе реализации результаты.

Методы управления проектами позволяют: определить цели проекта и пронести его обоснование; выявить структуру проекта; определить необходимые объемы и источники финансирования; подобрать исполнителей через процедуры торгов и конкурсов; подготовить и заключить контракты; определить сроки выполнения проекта, составить график его реализации, рассчитать необходимые ресурсы; рассчитать смету и бюджет проекта; планировать и учитывать риски; обеспечить контроль за ходом выполнения проекта.

Разработка программного обеспечения в большинстве случаев должна рассматриваться как коллективный труд специалистов, направленный на удовлетворение потребности пользователей в автоматизации их деятельности. Как и любой другой коллективный труд, она требует организации, в частности — *управления*. Это процесс, порою длительный, связывающий производственными и иными отношениями тех, кого в той или иной степени можно рассматривать в качестве производителей программы. Как и любой труд, тесно связанный с неоднозначными потребностями тех, кто будет использовать продукты труда, необходимым элементом *разработки программ* является решение задач изучения пользователей, с одной стороны, а с другой — обеспечения обратной связи с ними, направляющей производство. Это составляющие, из которых формируются главные задачи управления производством программ. Чаще всего решение таких задач осуществляется руководителем, или, как принято говорить, ***менеджером проекта***.

Понятие "*менеджер проекта*" не обязательно соотносится с конкретной персоной, отвечающей за управление производством

программной системы в целом. В небольшом проекте такое единоначалие чаще всего оправданно: оно позволяет концентрировать все нити управления, исключает проблемы внутреннего для проекта согласования противоречий, обеспечивает централизованную ответственность за проект перед теми, кто заинтересован в его выполнении. Однако по мере перехода к более масштабным проектам менеджерские обязанности становится невозможно концентрировать в одних руках. Обычно в таких случаях поступают в соответствии с одной из двух схем организации производства.

- Первая схема — это образование ***службы менеджера***, состоящей из его помощников, которым он может поручать различные задания, освобождая себя от рутины постоянного контроля. Этот путь, по существу, является лишь паллиативом.
- Для еще более сложных проектов появляется необходимость следовать второй схеме, т.е. образовывать ***группу менеджеров***, ответственных за разные разграниченные сферы проекта.

В этой схеме централизация достигается путем назначения главного *менеджера проекта*, который делегирует полномочия менеджерам по направлениям.

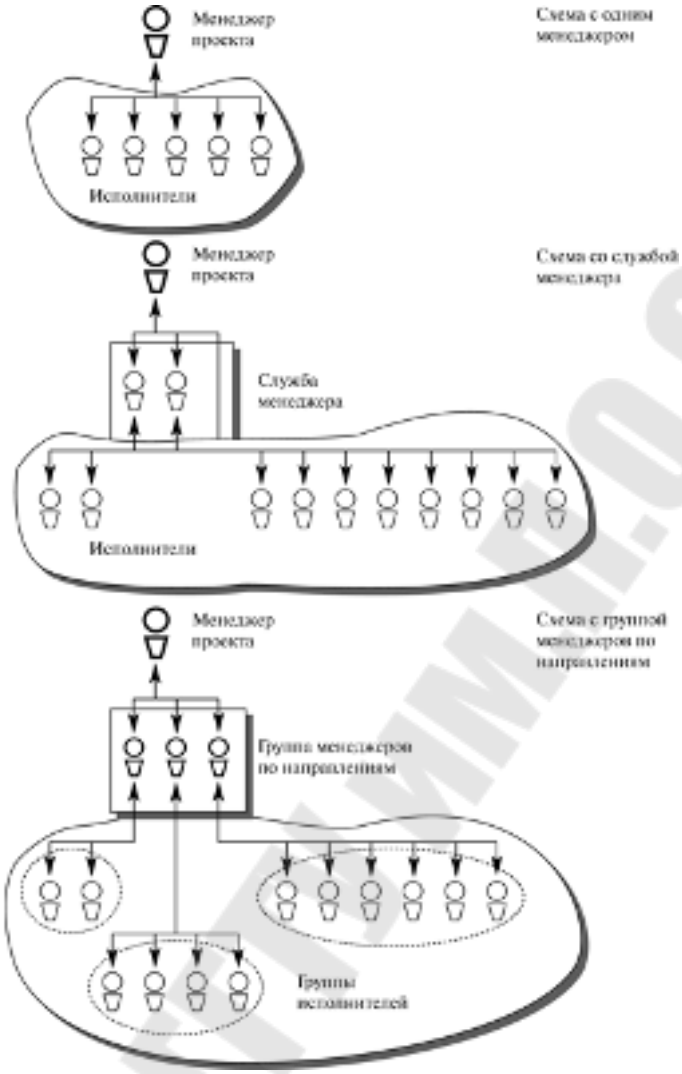


Рис. 1.1. Менеджмент проекта

Рис. 1.1 иллюстрирует три схемы организации менеджмента проекта. Здесь стрелки обозначают связи участников реализации проекта, обусловленные их взаимодействием с менеджером, жирность контура значков отражает менеджерские обязанности сотрудников. Как видно из рисунка, в схеме со службой менеджера помощники по своему статусу являются обычными работниками, тогда как при *делегировании полномочий* менеджеры по направлениям получают соответствующие полномочия в своих сферах ответственности (обозначены пунктирными овалами).

Делегирование можно считать основным инструментом разделения труда в проекте (не только в случае *менеджмента*), когда есть ответственность за некоторую функцию (работу и др.), но для ее выполнения нет собственных ресурсов, а потому приходится прибегать к помощи. Собственно говоря, различие схем работы

менеджера связано с использованием механизмов поручений или ***делегирования***.

Для небольших групп возможна следующая организация работ: обязанности главного менеджера распределяются по команде разработчиков, которая за счет внутренних механизмов решает, как планировать работы, как их распределять и контролировать. Это характерно для так называемого подхода быстрой *разработки (agile development) программных систем*, объединившего в себе несколько методологий программирования, которые отказываются от многих принципов традиционного *менеджмента*. Наиболее ярким примером подхода быстрой разработки является экстремальное программирование. Ему и другим методам данного направления мы уделим внимание в дальнейшем, а пока лишь отметим, что здесь, как и в других случаях, менеджерские задачи не исчезают, как могло бы показаться на первый взгляд. Просто форма и методы их решения приобретают другой характер.

По ряду причин может оказаться целесообразным сочетание схем организации *менеджмента* проекта. В результате появляются, например, коллективы, в которых главный менеджер берет на себя функции менеджера определенного направления, тем самым он отвечает и за весь проект в целом, и за некоторую его часть.

Прием определения абстрактного менеджера следует считать методическим. В дальнейшем изложении он называется определением ***абстрактного действующего лица проекта*** и применяется к изучению задач, связанных с выполнением проекта, без привязки их деятельности к конкретным персоналиям или групповым ролям.

В работе менеджера всегда присутствуют и неразрывно связаны друг с другом два аспекта:

- *управление проектом* как деятельность, продвигающая процесс производства к определенным целям,
- *руководство людьми*, *участниками разработки*.

С организационной точки зрения в *разработке программного обеспечения* можно выделить три варианта целей, определяющие деятельность *менеджера*:

- **производство программ не для продажи**, напрямую не связанное с получением дохода.
- **производство *рыночного продукта***, обеспечивающего прибыль за счет распространения (продажи) получаемых результатов.

• **разработка ведется под заказ**, когда все производство программы, от стадии замысла до передачи в эксплуатацию, финансируется внешними по отношению к разработчикам, но весьма заинтересованными агентами, обычно называемыми заказчиками.

Главная и постоянная задача *менеджмента разработки программного обеспечения* — продвижение проекта к обозначенным в начале его развития результатам. Если оставить в стороне приемы и методы, с помощью которых достигается решение этой задачи, то она сводится к распределению и контролю эффективного использования имеющихся ресурсов проекта: времени, финансов, технических средств и производственного потенциала работников. Множественность критериев, необходимость согласования интересов участников проекта и его заказчиков, разнообразие видов деятельности, составляющих развитие проекта, — вот тот организационный и производственный контекст, который вынужден учитывать менеджер.

Характерной особенностью *разработки программного обеспечения* является стремление к *переиспользованию* ранее созданных программных компонентов. Задачи *переиспользования* — это:

• во-первых, определение программных продуктов или каких-либо иных изделий и инструментов, имеющихся в распоряжении разработчиков, использование которых могло бы способствовать прогрессу развития проекта,

• во-вторых, выявление компонентов данного проекта, которые было бы полезно задействовать в других разработках.

Второе дополнительное направление — это задача *распространения построенного программного продукта*. Если с самого начала не рассматривать ее и относить распространение лишь к этапам, следующим за разработкой, есть опасность сделать не то, что нужно потребителю продукции, и выпустить из рук рычаги, с помощью которых можно влиять на сферу возможного применения создаваемых программных продуктов.

С точки зрения управления ***участники проекта*** — это абстрактные действующие агенты, которые выполняют заданные функции. Здесь под функцией понимается некое действие, при выполнении которого потребляются определенные ресурсы и производится определенный результат. Функциональный взгляд на *участников разработки проекта* делает их взаимозаменяемыми,

обезличенными в пределах компетенции, соответствующей выполнимости функции. Он приводит к понятию роли, назначаемой работнику для выполнения соответствующих обязанностей.

Ключевым качеством коллектива, определяющим его успешность, является слаженность. В идеальном коллективе все понимают друг друга с полуслова, есть взаимопонимание и уважение, не происходят или сведены к минимуму внутренние конфликты и противоречия. В реальности менеджеру редко приходится иметь дело с таким коллективом, а значит, необходимы меры, способствующие не только росту индивидуальной квалификации работников, но и дееспособности формируемой команды в целом. Эти меры следует рассматривать в качестве задач менеджера как руководителя коллектива.

Таким образом, при обсуждении задач менеджера в разработке программных изделий мы сталкиваемся с тремя взаимосвязанными *направлениями его деятельности*.

- Первое направление — это те *функции*, которые необходимо выполнять для успешного развития проекта. Здесь следует выяснить, какие роли сотрудников требуются для данного проекта.
- Второе направление — планирование и контроль хода проекта в соответствии с *жизненным циклом создаваемого программного обеспечения*.
- Наконец, третье направление определяется кругом задач формирования коллектива и, в частности, кадровым обеспечением проекта.

Эти направления подробно рассматриваются в дальнейших лекциях предлагаемого курса, рассчитанного на широкий круг специалистов в области *разработки программного обеспечения*.

Вопросы для самоконтроля:

1. Назовите 3 варианта целей, определяющих деятельность менеджера
2. Назовите основные задачи менеджера по разработке программных проектов
3. Назовите основные направления деятельности менеджера по разработке программных проектов

2. ФУНКЦИОНАЛЬНЫЕ РОЛИ В КОЛЛЕКТИВЕ РАЗРАБОТЧИКОВ. КЛЮЧЕВЫЕ РОЛИ И ОПРЕДЕЛЕНИЕ КАДРОВОГО РЕСУРСА

2.1. Функциональные роли в коллективе разработчиков

Для целенаправленного выполнения проекта должен быть выполнен ряд работ, различных как по своему назначению, так и по квалификационным требованиям, предъявляемым к *разработчикам*. Иными словами, в ходе развития проекта командой *разработчиков* выполняются те или иные функции.

Функции, выполняемые *разработчиками*, — понятие неформализованное. В разных проектах оно может обретать свое содержание. Тем не менее *типовые функции*, которые предполагают практически все программные проекты, можно перечислить. Так, в любом программном проекте есть функции кодирования, т.е. записи программы на алгоритмическом языке по имеющимся спецификациям, анализа требований, т.е. выявления истинной потребности в создаваемой программе, тестирования и отладки. Далее мы рассмотрим эти и другие функции *разработчиков*, связанные с проектом, а пока лишь заметим, что в рамках деятельности менеджера любого проекта необходимо организовать *распределение функций* проекта между исполнителями. Вполне естественно считать эти действия одной из функций менеджера. В результате ее выполнения члены команды, выполняющей проект, начинают играть соответствующие роли.

Таким образом, в рамках любого проекта возникает задача повышения квалификации сотрудников. Для различных схем ведения проектов эта задача решается по-разному. Крайнюю точку зрения на проблему соответствия квалификации работников требованиям проекта отражает идея ***экстремального программирования***, когда используется неформальная организация группы исполнителей проекта без четкого распределения ролей, а значит, и обязательств сотрудников. В этом случае провозглашается принцип, когда каждый в группе делает то, что он умеет делать лучше всего. И хотя все функции, которые должны выполняться, остаются, создается впечатление, что в группе исполнителей проекта исчезают роли. В результате возможны пробелы в разработке, в частности при анализе и декомпозиции проектируемой системы. Чтобы этого избежать,

разработчики должны понимать, какую абстрактную роль они исполняют в каждый конкретный момент, выполнение каких проектных функций необходимо сейчас, как связаны между собой работы, как должны распределяться ресурсы. Словом, они должны обращать внимание на выполнение распределенных по группе менеджерских функций. И даже в этом случае схему *экстремального программирования* можно рекомендовать лишь для слаженных групп исполнителей с высоким уровнем коллективной ответственности.

В *модели проектной группы концепции Microsoft Solution Framework (MSF)* предлагается образовывать небольшие мобильные коллективы как атомарные производственные единицы с общей ответственностью за выполняемые задания — так называемые проектные группы. Такие группы строятся как многопрофильные команды, члены которых распределяют между собой ответственность и дополняют *области компетентности* друг друга. Группа состоит не более чем из 10 человек. Все они считаются обладающими сходным уровнем профессиональной подготовки, хотя и в разных областях индивидуальной специализации. Провозглашается равноправие членов группы и коллективная ответственность за выполняемые задания: проектная группа — команда равных. Все это позволяет сохранять внутри группы неформальные отношения.

Вместо понятия роли для группы в целом определяются ***ролевые кластеры***, которые заполняются точно так же, как происходит распределение ролей. В то время как за успех проекта ответственна вся команда, каждый из ее *ролевых кластеров*, определяемых моделью, ассоциирован с одной из проектных целей и работает над ее достижением. В данной модели именно эти цели задают роли *разработчиков*, которые определяются кластерами. В терминологии MSF используется понятие *области компетенции*, или *области функциональной специализации* (functional area), обозначающее ту или иную роль, которую выполняет кластер группы в проекте. Принципиальное отличие распределения исполнителей по *ролевым кластерам* от распределения ролей заключается лишь в том, что ответственность за это несет не *менеджер проекта*, а сама группа. Менеджер проекта выдает задания и контролирует их выполнение лишь в целом для группы, не вмешиваясь в ее работу.

Определено шесть *ролевых кластеров*, которые соответствующим образом структурируют проектные функции *разработчиков* (рис. 2.1).

Рис. 2.1. Ролевые кластеры модели проектной групп MSF

- ***Управление продуктом*** (product management). Ключевая цель кластера — обеспечивать удовлетворение интересов *заказчика*. Для ее достижения кластер должен содержать следующие области компетенции:
  - планирование продукта,
  - планирование доходов,
  - представление интересов *заказчика*,
  - маркетинг.
- ***Управление программой*** (program management). Задача — обеспечить реализацию решения в рамках ограничений проекта, что может рассматриваться как удовлетворение требований к бюджету проекта и к его результату. *Области компетенции* кластера:
  - управление проектом;
  - выработка архитектуры решения;
  - контроль производственного процесса;
  - административные службы.
- ***Разработка*** (development). Первостепенной задачей кластера является построение решения в соответствии со спецификацией. *Области компетенции* кластера:
  - технологическое консультирование;
  - проектирование и осуществление реализации;
  - разработка приложений;
  - разработка инфраструктуры.
- ***Тестирование*** (test). Задача кластера — одобрение выпуска продукта только после того, как все дефекты выявлены и устранены. *Области компетенции* кластера:
  - разработка тестов;

- отчетность о тестах;
- планирование тестов.

• ***Удовлетворение потребителя*** (user experience). Цель кластера — повышение эффективности использования продукта. *Области компетенции* кластера: — общедоступность (возможности работы для людей с недостатками зрения, слуха и др.);

- интернационализация (эксплуатация в иноязычных средах);
- обеспечение технической поддержки;
- обучение пользователей;
- удобство эксплуатации (эргономика);
- графический дизайн.

• ***Управление выпуском*** (release management). Задача кластера — беспрепятственное внедрение и сопровождение продукта. *Области компетенции* кластера:

- инфраструктура (infrastructure);
- сопровождение (support);
- бизнес-процессы (operations);
- управление выпуском готового продукта (commercial release management).

Мы придерживаемся ролевой структуры проекта, которая предложена Центром объектно-ориентированной технологии компании IBM. Эта структура включает достаточно полный перечень типичных ролей, согласованный со многими реальными дисциплинами развития программных проектов. В то же время она представляет роли *разработчиков* в организационном контексте, т.е. рассматривает не только *разработчиков*, но и тех, кто, не участвуя в проекте в качестве исполнителей, оказывает влияние на постановку задач проекта, на выделение ресурсов и обеспечение осуществимости развития работ.

• ***Заказчик*** (Customer) — реально существующий (в организации, которой подчинена команда, или вне ее) инициатор разработки или кто-либо иной, уполномоченный принимать результаты (как текущие, так и окончательные) разработки.

• ***Планировщик ресурсов*** (Planner) — выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку, а также развивает и направляет план выполнения проекта с точки зрения организации.

• ***Менеджер проекта*** (Project Manager) — отвечает за развитие проекта в целом, гарантирует, что распределение заданий и ресурсов

позволяет выполнить проект, что работы и предъявление результатов идут по графику, что результаты соответствуют требованиям. В рамках этих функций *менеджер проекта* взаимодействует с *заказчиком* и *планировщиком ресурсов*.

• ***Руководитель команды*** (Team Leader) — производит техническое руководство командой в процессе выполнения проекта. Для больших проектов возможно привлечение нескольких руководителей подкоманд, отвечающих за решение частных задач.

• ***Архитектор*** (Architect) — отвечает за проектирование архитектуры системы, согласовывает развитие работ, связанных с проектом.

• ***Проектировщик подсистемы*** (Designer) — отвечает за проектирование подсистемы или категории классов, определяет реализацию и интерфейсы с другими подсистемами.

• ***Эксперт предметной области*** (Domain Expert) — отвечает за изучение сферы приложения, поддерживает направленность проекта на решение задач данной области.

• ***Разработчик*** (Developer) — реализует проектируемые компоненты, владеет и создает специфичные классы и методы, осуществляет кодирование и автономное тестирование, строит продукт. Это широкое понятие, которое может подразделяться на специальные роли (например, *разработчик* классов). В зависимости от сложности проекта команда может включать различное число *разработчиков*.

• ***Разработчик информационной поддержки*** (Information Developer) — создает документацию, сопровождающую продукт, когда выпускается версия. Включаемые в нее инсталляционные материалы, равно как ссылочные и учебные, а также материалы помощи предоставляются на бумажных и машинных носителях. Для сложных проектов возможно распределение этих задач между несколькими *разработчиками информационной поддержки*.

• ***Специалист по пользовательскому интерфейсу*** (Human Factors Engineer) — отвечает за удобство применения системы. Работает с *заказчиком*, чтобы удостовериться, что пользовательский интерфейс удовлетворяет требованиям.

• ***Тестировщик*** (Tester) — проверяет функциональность, качество и эффективность продукта. Строит и исполняет тесты для каждой фазы развития проекта.

• ***Библиотекарь*** (Librarian) — отвечает за создание и ведение общей библиотеки проекта, которая содержит все проектные рабочие продукты, а также за соответствие рабочих продуктов стандартам.

Взаимодействие менеджера с *заказчиком*, *планировщиком* и другими *инициаторами работ* — прямая обязанность *менеджера проекта*. Таким образом, для него закреплен круг функций, которые являются внешними по отношению к работам над проектом.

Выполнение *внешних функций* создает лишь условия для разработки. Как видно из перечня ролей, другой круг функций, возлагаемый на менеджера, связан с взаимодействиями с действующими лицами из команды *разработчиков* проекта. Это ***внутренние функции менеджера***.

В зависимости от проекта и условий его выполнения роли участников проекта могут совмещаться. Предельный случай — программист разрабатывает проект для себя (по собственному заказу), сам планирует распределение ресурсов (сроки выполнения работы, использование вычислительной техники и др.), сам принимает проектные решения (управляет и руководит собою) и сам же занимается разработкой, экспертизой и обслуживанием.

В большинстве случаев *заказчик* и *планировщик ресурсов* являются действительно внешними по отношению к проекту действующими лицами, а потому *совмещение этих ролей* с другими — нечто экзотическое. Тем не менее, роль *заказчика* как члена коллектива *разработчиков*, аккумулирующего точки зрения всех *инициаторов работ*, весьма полезна. В частности, подход *экстремального программирования* считает это обязательным, чтобы развитие проекта всегда гарантированно было направлено в сторону, нужную пользователям.

Менеджер проекта по своему назначению является выделенным в команде. Он берет на себя взаимодействие с *заказчиком* и *планировщиком ресурсов*, с одной стороны, а с другой — распределяет работы среди членов команды. Последнее означает, что он должен обладать полной информацией о декомпозиции проекта. Как следствие, совмещение его роли с ролью *архитектора* проекта является весьма желательным, а потому, довольно частым. Единственным условием для такого совмещения является требование четко знать, когда и чьи функции выполняет данное действующее лицо.

Вполне возможно продуктивное *совмещение ролей руководителя команды* и *архитектора* — это дает ощутимые результаты, когда команда достаточно крепко связана со своим руководителем, например, по предшествующим работам, но при условии не слишком высокой сложности задач декомпозиции для данного проекта.

Совмещение ролей *руководителя команды* и менеджера допустимо, но лишь тогда, когда осознается и учитывается противоречивость целевых установок этих ролей: *руководитель команды* действует в условиях, которые формируются менеджером.

Нежелательно *совмещение ролей руководителя команды* и *проектировщика* какой-либо подсистемы. И это обусловлено противоречивостью их ролевых интересов.

Если используется *перекрестное совмещение ролей руководителя команды* и *проектировщика подсистемы*, то возникает еще одно противоречие их ролевых интересов: у руководителя могут быть предпочтения в пользу "своего" компонента и, как следствие, возможен дисбаланс проекта в целом в пользу выделенных его составляющих, который придется компенсировать дополнительными усилиями менеджера.

По тем же причинам не допускается *совмещение ролей* менеджера и *разработчика*. Здесь запрет более жесткий, поскольку контрольные функции менеджера несовместимы с исполнительскими задачами *разработчика*. Даже в тех случаях, когда у менеджера остается свободное время после выполнения своих прямых обязанностей, ему не следует "помогать" *разработчикам*. Лучше занять себя другим делом, в частности выступить в роли *разработчика* другого проекта.

В то же время *совмещение ролей различных разработчиков* — обычное дело для больших проектов. Оно является частью распределения работ по исполнителям. Решают эту задачу совместно *руководитель команды* и менеджер, когда основные архитектурные положения утверждены. Способы решения зависят от того, как формируется команда. Если разработка поручается готовой команде, то возрастает роль руководителя, а менеджер лишь утверждает его предложение. Когда команда создается для проекта, растет удельный вес менеджерских функций в подборе кадров для работ. Но в любом случае при *совмещении ролей различных разработчиков* нужно учитывать уровень квалификации работников, их предпочтения и

другие индивидуальные особенности. Необходимо знать, что, выделяя одному действующему лицу несколько работ, следует стремиться к однородности заданий и указывать их сравнительные приоритеты.

Допустимы и другие *совмещения ролей*. Так, довольно часто создание документации распределяется между всеми исполнителями проекта, а на *руководителя команды* и менеджера возлагается задача интеграции документов. Для обозримых проектов функции библиотекаря можно поручить одному из *разработчиков*, соответственно скорректировав его индивидуальное задание (см. принцип 3). Специалистом по пользовательскому интерфейсу вполне может быть, например, менеджер, поскольку именно он осуществляет контакты с *заказчиком* (в данной ситуации *заказчик* либо сам является пользователем, либо выступает как представитель пользователя программного изделия).

По поводу *тестировщиков* необходимы некоторые разъяснения. Следует различать два вида тестирования: тестирование модулей, автономное по своей сути, и тестирование предоставляемых функций, которое может быть и комплексным (проверка совместного выполнения функций), и автономным, когда проверяется работоспособность отдельного аспекта системы. В первом случае задачи *тестировщика* невозможно выполнить без знания не только назначения, но и структуры проверяемого модуля. Лучше всего в этом разбирается *разработчик* модуля, а значит, именно ему этот модуль и отлаживать. Автономное тестирование второго вида, по существу, есть другая сторона проверки работоспособности модуля, реализующего определенную функцию. Во многих случаях даже нет нужды различать его и тестирование модуля. Поэтому и здесь разумно говорить о деятельности *разработчика*. В обоих случаях самотестирование для *разработчика* есть не дополнительная роль в проекте, а часть его профессиональных обязанностей в рамках автономной отладки.

Другой метод, нисколько не противоречащий априорному тестированию, — перекрестное тестирование, когда *разработчики* независимых компонентов проекта тестируют функциональность друг друга. Здесь можно говорить о *перекрестном совмещении ролей разработчиков* и *тестировщиков*, поскольку для выполнения перекрестного тестирования нужно различное представление об испытываемом модуле.

В качестве итога обсуждения совместимости ролей для действующих лиц проекта в табл. 2.1 приводятся краткие характеристики *совмещения ролей*, которые можно рассматривать как рекомендации для менеджмента. Разумеется, нельзя абсолютизировать эти характеристики для любых проектов. Не стоит жестко фиксировать распределение ролей и в одном проекте. Более того, распределение ролей можно рассматривать в качестве одного из инструментов, с помощью которого менеджер может руководить коллективом. Если этим инструментом пользоваться умело, то удается добиться наиболее эффективного разделения труда, соответствующего индивидуальным особенностям участников проекта.

Таблица 2.1

Совместимость ролей

Роли	**Характеристика совмещения ролей**
Менеджер и архитектор	Желательно
Менеджер и руководитель команды	Противоречиво
Руководитель команды и архитектор	Возможно
Руководитель команды и проектировщик подсистемы	Нежелательно
Менеджер и разработчик	Не допускается
Для различных разработчиков	с ограничениями
Создание документации (все сотрудники)	Успешно распределяется
Специалист по интерфейсу и менеджер	Разумно
Эксперт предметной области и менеджер	Зачастую разумно
Специалист по интерфейсу и эксперт предметной области	Редко бывает эффективно
Эксперт предметной области и разработчик	Бывает полезно
Специалист по интерфейсу и разработчик	Часто полезно
Библиотекарь и один из разработчиков	Допустимо
Тестировщики и другие члены команды	Перекрестно
Эксперт предметной области, тестировщик	Оправданно

2.2. Ключевые роли коллектива разработчиков и задача определения кадровых ресурсов проекта

Руководство коллективом разработчиков — постоянная задача менеджера. Очень часто она выходит за пределы проекта, поскольку опытный менеджер, когда берется за проект, всегда имеет в виду команду, с которой ему будет комфортно работать, которая справится с заданием вполне гарантированно. Подобные команды не складываются сразу, а формируются в процессе выполнения проектов. В конечном счете за руководство стоит браться только тогда, когда есть шанс в процессе выполнения работ получить такую команду.

В ходе априорного распределения ресурсов менеджер определяет кандидатуры на *ключевые роли коллектива разработчиков*. Само понятие ключевой роли указывает на то, что от сотрудников, которым такие роли назначены, в наибольшей степени зависит успех проекта. Какие роли в проекте являются ключевыми, во многом зависит от специфики разработки.

Намного лучше, когда лидерство сотрудников обнаруживается или заранее известно на ранней стадии проекта. Это позволяет сразу правильно выстроить административную структуру. Тем не менее и здесь есть свои подводные камни. Не все *лидеры* готовы или хотят брать на себя управленческие обязательства. В этом случае лучше с самого начала назначить администратора, который освободит фактического *лидера* от перманентных административных проблем. Другая проблема — соперничество за лидерство в коллективе, что является потенциально рискованной для проекта ситуацией. Если такое случается, а еще лучше, когда соперничество только зарождается, сразу же разделить сферы ответственности конкурентов.

Есть еще одна причина, по которой ситуация с менеджером в качестве *лидера* может отрицательно сказываться на результативности выполнения проекта: в таком случае ему неизбежно придется вникать в детали производственного процесса, которые не соответствуют менеджерской функции распределения ресурсов, работ и контроля.

Наличие единственного *лидера* в группе, на которого может положиться менеджер проекта, — одна из главных предпосылок формирования продуктивно работающего коллектива. Но если появляется *противодействующий лидер*, ситуация в проекте

становится крайне неустойчивой. И менеджеру нужно всячески избегать ее: распознавать противодействие как можно раньше, быть терпимым и терпеливым при обсуждении вариантов решений даже тогда, когда кажется, что решение лежит на поверхности. Следует подчеркнуть, что в такой ситуации "хирургическое вмешательство" способно разрушить коллектив со всеми вытекающими отсюда последствиями. Борьба с *лидером*, особенно со скрытым неформальным *лидером*, к тому же заметная членам команды, также плохое средство. Она может быть действенной только в тех случаях, когда руководитель уверен в своей победе без административных воздействий. По сути, такая борьба должна рассматриваться в рамках мер по смене *лидера* команды. И наиболее трудной она оказывается тогда, когда *противодействующий лидер* выполняет одну из *ключевых ролей* в проекте.

Следующий перечень ключевых ролей *характеризует наиболее типичные ситуации для программных проектов:*

- архитектор проекта;
- проектировщики подсистем;
- руководители команд разработки подсистем;
- специалист по пользовательскому интерфейсу;
- эксперт предметной области

Безусловно, *ключевой ролью* в проекте является лишь архитектор. В связи с этим уместно отметить, что провозглашаемый сторонниками экстремального программирования принцип программирования без архитектора может означать только одно: его роль явно или неявно распределяется между членами команды. При таком подходе говорят, что все разработчики команды занимают *ключевые роли* с самого начала проекта. Поэтому ясно, почему команды, действующие по схемам экстремального программирования, открываются для привлечения дополнительных кадров довольно редко. При необходимости выполнять работы, выходящие за рамки компетенции сотрудников, они чаще всего прибегают к субподряду.

- Во-первых, менеджер может **заранее знать возможных кандидатов (хорошо себя зарекомендовавших сотрудников фирмы, с которыми** менеджер уже имел дело либо знает по чужим работам). Это самый комфортный вариант для менеджера, который вполне может использовать личные, иногда лишь интуитивные представления о сотрудниках для выбора кандидатов. Следует,

однако, иметь в виду, что ни в коем случае нельзя подменять знание о человеке как о работнике сведениями личного характера (приятельские отношения, коммуникабельность и др.).

• Во-вторых, менеджер может **подбирать кандидатов из числа сотрудников фирмы**, с которыми он еще не имел дело. В этом случае ему не обойтись без изучения послужного списка сотрудника, *собеседований* и прочих способов получения сведений о человеке.

• Третий вариант, на который стоит рассчитывать, если возможности предыдущих вариантов исчерпываются, — это **принять на работу нового сотрудника**. Ситуация почти такая же, как в предыдущем случае, но несколько хуже: про сотрудника фирмы довольно просто получить объективную информацию, тогда как, нанимая нового человека, приходится довольствоваться сведениями из документов и результатами *собеседований*.

Решение *задачи подбора кадров* для проекта зависит от условий, в которых действует менеджер. Ситуации могут быть следующие:

1. У менеджера есть коллектив потенциальных исполнителей, готовый приступить к работе над проектом.
2. Менеджерский коллектив потенциальных исполнителей недостаточен: среди членов команды нет сотрудников, которые обладают нужной квалификацией.
3. В поле зрения менеджера есть независимые потенциальные исполнители, из которых можно сформировать коллектив для работы над проектом.
4. Менеджеру приходится прибегать к найму рабочей силы со стороны.

Таким образом, менеджмент проекта рассматривается как обеспечение поставок продукта, разработка которого требует выполнения определенного объема работ (область действия) с привлечением затрат, не выходящих за определенные пределы, укладывающаяся в заданные рамки времени и удовлетворяющая приемлемому уровню качества. Это широко известный "***треугольник менеджмента проектов***", представленный на рис. 2.2.

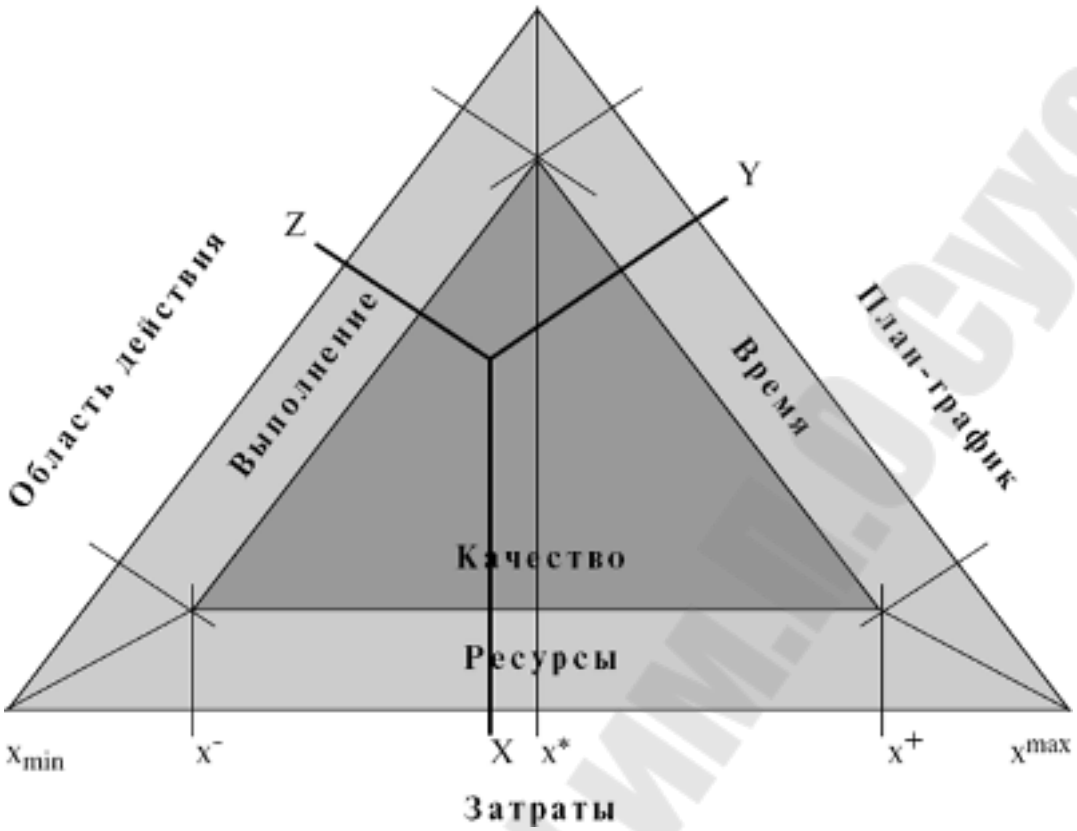


Рис. 2.2. Треугольник менеджмента проектов

Он интерпретируется как задача менеджера, формулируемая следующим образом. Нужно уравновешивать производительность работ проекта (область действия), время (план-график поставок) и расход ресурсов (затраты), удовлетворяя требованиям качества. По разным причинам соблюдать баланс по всем параметрам чаще всего не удается, а потому менеджер вынужден выбирать в качестве первичной цели только один или два параметра, ставя остальные в зависимость от них. Это действие приводит к такой интерпретации рисунка: "треугольник хорошо-быстро-дешево — выбери два из них".

В ходе управления должно быть обеспечено рациональное использование всех ресурсов, необходимых для создания и функционирования изделия. Процесс поиска оптимального соотношения между различными ресурсами при подготовке к реализации проекта называется *принципом совмещения методологий*.

2.3. Система мотивации персонала

Система мотивации персонала (система вознаграждений) включает в себя все, что наемный работник может ценить и желать и что работодатель в состоянии или желает предложить в обмен на вклад наемного работника в выполнение организацией ее миссии, -

так определяет систему вознаграждений автор классического труда "Компенсационный менеджмент" Р.И. Хендерсон.

Ряд авторов принципиально разделяют понятия "мотивация персонала в организации" и "стимулирование персонала в организации" персонала. Под "мотивацией персонала в организации " при этом понимаются внутренние мотивы, определяемые потребностями человека и побуждающие его к действию. Под "стимулированием" - все внешние воздействия, которые работодатель оказывает на работника с целью побудить его к выполнению поставленных задач. С этой точки зрения то, что обычно принято называть "системой мотивации", является "системой стимулирования". Поскольку в практике в большинстве случаев не делается существенного разграничения между этими понятиями, мы будем употреблять их как синонимы.

Важнейшими признаками системы мотивации персонала являются то, что она определяется целями бизнеса организации и задачами, которые перед компанией стоят, а также ее взаимосвязь с другими системами, определяющими функционирование организации – системой финансового планирования и учета, системой продаж и др.

В системе мотивации персонала в организации выделяют компенсационные и некомпенсационные компоненты, влияющие на повышение мотивации сотрудников. В подсистему компенсаций входят все вознаграждения, которые можно классифицировать как монетарную и натуральную оплату. К монетарной оплате относят все денежные и эквивалентные денежным (например чеки, кредитные карты) формы оплаты. Натуральная оплата – это товары и услуги, которые используют вместо денег. Все иные вознаграждения образуют некомпенсационную систему.

К измерениям компенсационной системы Р.И. Хендерсон относит:

1. Плату за работу и производительность.
2. Продолжение выплат при нетрудоспособности.
3. Отсроченный доход.
4. Охрану здоровья от несчастного случая и пр.
5. Плату за нерабочее время.
6. Продолжение выплат при утрате работы.
7. Продолжение выплат на супруга (семью).
8. Оплата, эквивалентная доходу.

К измерениям некомпенсационной системы относятся:

1. Повышение чувства собственного достоинства и удовлетворения от работы.

2. Улучшении физического здоровья, интеллектуальный рост и эмоциональное совершенствование.

3. Поощрение конструктивных социальных взаимосвязей с коллегами по работе.

4. Конструирование заданий, требующих адекватного внимания и усилий.

5. Предоставление достаточных ресурсов для выполнения порученных работ.

6. Гарантия достаточности контроля задания с целью удовлетворения личных запросов.

7. Предложение поддерживающего лидерства и менеджмента.

Повышение мотивации сотрудников является одной из самых сложных задач менеджмента компании. Поэтому внедрение системы мотивации персонала организации – это сложный и длительный проект, требующий существенных временных и финансовых вложений. Результатом внедрения системы мотивации персонала, поддерживающей достижение целей бизнеса, является существенный рост финансовых показателей компании (выручка, прибыль), выход на новый уровень клиентов, а также качественное изменение персонала организации.

Основные этапы внедрения системы мотивации персонала в организации:

1. Прояснение целей бизнеса.

2. Формирование рабочей группы.

3. Формирование плана разработки и внедрения системы мотивации.

4. Презентация плана топ-менеджерам, его утверждение.

5. Создание системы фиксированного вознаграждения (тарифная сетка, система грейдов и пр.).

6. Создание системы премирования.

7. Создание некомпенсационной системы мотивации.

8. Подготовка регламентирующих документов.

9. Презентация системы руководителям и сотрудникам.

10. Внедрение системы в тестовом режиме, внесение необходимых корректировок.

11. Внедрение системы мотивации в компании.

12. Мониторинг результатов, внесение изменений.

На практике целесообразно внедрять составляющие системы мотивации персонала последовательно, а не все сразу. Поскольку мотивационный аспект является одним из самых значимых для сотрудников компании, его изменения всегда являются для них стрессом и требуют системной работы по разъяснению механизмов действия новой системы оплаты и нематериальной мотивации.

Вопросы для самоконтроля:

1. В чем заключается идея экстремального программирования?
2. Назовите ролевые кластеры, структурирующие проектные функции разработчиков.
3. Назовите участников ролевой структуры проекта, предложенной компанией IBM.
4. Какие роли в коллективе разработчиков целесообразно совмещать?
5. Какие роли в коллективе разработчиков совмещать крайне нежелательно?
6. Какие типы тестирования программного обеспечения существуют, и с какой ролью целесообразно совмещать роль тестировщика?

3. ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ИЗДЕЛИЯ, ТРАДИЦИОННЫЕ МОДЕЛИ

3.1. Понятие жизненного цикла программного изделия

Очевидно, что функции, выполняемые разработчиками проекта, в ходе его *развития* претерпевают изменения, как, впрочем, и сам проект. Сначала он существует в виде заявки на *разработку*, затем — как функциональные и технические требования, далее — как спецификации разрабатываемого изделия, набор программных модулей, скомпонованная из модулей система и т.д. Этот перечень можно рассматривать как один из примеров *модели жизненного цикла программного изделия*, т.е. представления эволюции *разработки* и последующего *использования* программной системы.

Жизненный цикл следует рассматривать как основу деятельности менеджера программного проекта: с ним связываются и цели проекта — окончательные и промежуточные, распределение и контроль расходования ресурсов, а также все другие аспекты управления *развитием проекта*. Прежде всего, эта привязка обусловлена разбиением *производства* любой программы на этапы, которые ассоциируются с определенными видами работ или функций, выполняемых разработчиками в тот или иной момент *развития проекта*. Этапы характеризуются направленностью выполняемых функций на достижение локальных (для этапа) целей проекта. Необходимость отслеживания целей приводит к понятию контрольных точек — моментов *разработки*, когда осуществляется подведение промежуточных итогов, осмысление достигнутого и ревизия сделанных ранее предположений.

Таким образом, в рамках обсуждения менеджмента программных проектов вопросы *жизненного цикла* должны рассматриваться как первостепенные. В этой и последующих лекциях они разбираются в той мере, которой достаточно для самостоятельного изучения конкретных подходов и методов, рекомендуемых для применения при *производстве* программных систем.

Программы не подвержены физическому износу, но в ходе их эксплуатации обнаруживаются ошибки (неисправности), требующие исправления.

Ошибки возникают также от изменения условий *использования* программы. Последнее является принципиальным свойством программного обеспечения, иначе оно теряет смысл. Поэтому правомерно говорить о *старении программ*, правда, не о физическом, а о "моральном". Необходимость внесения изменений в действующие программы (как из-за обнаруживаемых ошибок, так и по причине развития требований) приводит, по сути дела, к тому, что *разработка* программного обеспечения продолжается после передачи его пользователю и в течение всего времени жизни программ. Деятельность, связанная с решением довольно многочисленных задач такой продолжающейся *разработки*, получила название ***сопровождения*** программного обеспечения

Первоначально понятие *жизненного цикла* рассматривалось как *цикл разработки*. Однако понимание того, что стоимость программного обеспечения включает издержки в течение всего времени жизни системы, а не только затраты на *разработку* или исполнение программ, привело к естественной трансформации исходного понятия *цикла разработки*. ***Жизненный цикл*** — это проекция пользовательского понятия "время жизни" на понятие разработчика "технологический цикл (***цикл разработки***)". Комбинацией этих понятий объясняется происхождение самого термина "*жизненный цикл программного обеспечения*".

Последовательное развитие проекта и итеративное наращивание

Существуют различные подходы к *моделированию жизненного цикла* программного изделия, отражающие те или иные аспекты *разработки* программ и связанной с ней деятельности. В рамках тематики настоящего курса следует использовать такие модели, которые в полной мере дают представление об управленческой деятельности. Кроме того, особое внимание уделяется тем моделям, которые хорошо согласуются с прогрессивными методами *разработки* программных систем и прежде всего рассчитаны на построение систем согласно методологии объектной ориентированности.

Традиционный подход к *разработке* программных систем предполагал, что технологичное *развитие проекта* возможно только тогда, когда предварительно собраны и проанализированы все требования к будущему изделию. В результате такого анализа, с одной стороны, выясняется истинное назначение системы, а с другой

— появляется полная информация, необходимая для разбиения проекта на части, допускающие независимую *разработку*, — ***декомпозиции проекта***. После *декомпозиции* автономно реализуются выделенные части — модули или подсистемы (в зависимости от сложности требуемого программного продукта), причем в том же стиле, т.е. с предварительным полным анализом требований и, возможно, с дальнейшей *декомпозицией* частей. Затем осуществляется сборка — компоновка модулей в единую систему (подсистему — для выделенных частей). В результате продукт, поставляемый для *использования*, появляется в конце *разработки* всех частей и их компоновки. Если для проекта удается создать такие условия, что можно выделить для анализа все требования, то это гарантирует качество результатов. Однако на практике такая ситуация встречается крайне редко.

В описании традиционного подхода явно просматривается стремление следовать методологической стратегии, которую мы назвали определением этапов проекта и ***последовательным его развитием***. По существу, это механический перенос методики решения инженерных задач в промышленном *производстве* на область программирования. И как мы только что могли убедиться, условия материальной деятельности, в которых возможен и даже необходим предварительный сбор всех требований к проекту здесь выполнить не получается. Как следствие, стратегия *последовательного развития проектов,* по крайней мере, нуждается в коррективе.

При *объектно-ориентированном* подходе к *проектированию* провозглашается принцип ***возвратно-поступательного развития***, или итеративного наращивания системы, суть которого состоит в следующем. На каждой фазе проекта строятся работоспособные продукты, развиваемые в дальнейшем путем обогащения функциональности и интерфейса, а не в жестких рамках предварительного технического описания в целом, построенного в ходе специального этапа конструирования, предусматриваемого в традиционных схемах. Как следствие, фазы *развития проекта* (в традиционном понимании этого слова) при выполнении отдельной итерации оказываются незавершенными, они дополняются (наращиваются) на последующих итерациях.

Таблица 3.1

Сравнение параметров схем разработки проектов

Традиционные схемы	Объектно-ориентированная схема
Полностью завершенные фазы проектирования и программирования	Итеративно наращиваемые возможности. Традиционные фазы распределяются по итерациям
Продукт в конце периода разработки	Рабочие продукты на каждой итерации
Техническое описание как итог конструирования	Рабочее описание, дополняемое на каждой итерации
Последовательная разработка	Возвратно-поступательная разработка
Модули действий, операций	Иерархии классов объектов
Структурная, пошаговая детализация	Наследование, переопределение, полиморфизм

3.2. Модели традиционного представления о жизненном цикле

- **Общепринятая модель**

Вероятно, самым распространенным поводом для обращения к понятию жизненного цикла является потребность в систематизации работ в соответствии с производственным процессом. Этому назначению хорошо соответствует так называемая ***общепринятая модель жизненного цикла*** программного обеспечения, согласно которой программные системы проходят в своем развитии две фазы:

- разработка,
- *сопровождение*.

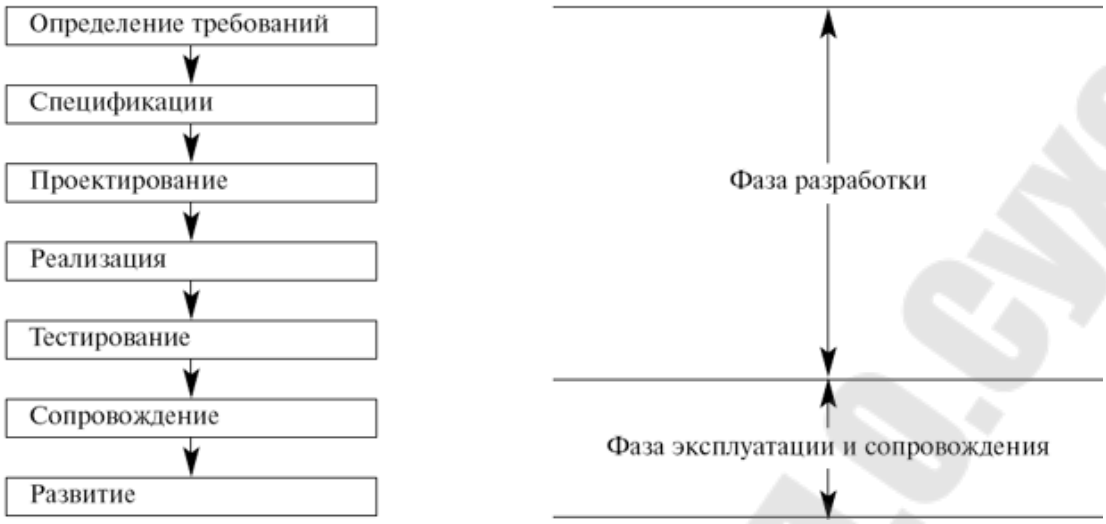


Рис 3.1. Схема общепринятой модели жизненного цикла проекта

Разработка начинается с *идентификации потребности* в новом приложении, а заканчивается передачей продукта разработки в эксплуатацию.

Первым этапом *фазы разработки* является постановка задачи и *определение требований*. Определение требований включает описание общего контекста задачи, ожидаемых функций системы и ее ограничений. На данном этапе заказчик совместно с разработчиками принимает решение о создании системы. В процессе согласования требований со стороны разработчиков в обсуждении участвует менеджер проекта, а также, возможно, планировщик как действующее лицо, заинтересованное в соблюдении условий ведения проекта с точки зрения фирмы и осведомленное о ее ресурсном потенциале. Принципиально, что на данном этапе самого проекта еще нет и можно говорить только о предпроектных работах, в которых участвуют исполнители, занимающие указанные выше роли.

Разработка проектных решений, отвечающих на вопрос о том, как должна быть реализована система, чтобы она могла удовлетворять специфицированным требованиям, выполняется на этапе *проектирования*. Поскольку сложность системы в целом может быть очень высока, главной задачей этого этапа является последовательная декомпозиция системы до уровня очевидно реализуемых модулей или процедур. Наиболее активная роль на данном этапе — архитектор, для которого декомпозиция системы есть главная задача в проекте.

На следующем этапе *реализации*, или кодирования каждый из модулей, выявленных при декомпозиции, программируется на наиболее подходящем для данного приложения языке. С точки зрения

автоматизации этот этап традиционно является наиболее развитым. Основные действующие лица этапа — руководитель команды и разработчики. Традиционно именно данный этап считали основой проекта в целом, что, как мы уже успели убедиться, не отражает современного взгляда на проект как на постоянно развивающийся артефакт. В обсуждаемой модели специально не выделяется этап сборки, который заключается в комплексации (интеграции) построенных и используемых модулей в систему. Считается, что это один из видов работ этапа *реализации*.

В *общепринятой модели фаза разработки* заканчивается этапом *тестирования* (автономного и комплексного) и передачей системы в эксплуатацию — следующие два этапа.

Фаза эксплуатации и *сопровождения* включает в себя всю деятельность по обеспечению нормального функционирования программных систем, в том числе фиксирование в скрытых во время исполнения программ ошибок, поиск их причин и исправление, повышение эксплуатационных характеристик системы, адаптацию системы к окружающей среде, а также, при необходимости, и более существенные работы по совершенствованию системы. Все это дает право говорить об эволюции системы. В связи с этим фаза эксплуатации и *сопровождения* разбивается на два этапа: собственно *сопровождение* и *развитие*. В ряде случаев на данную фазу приходится большая часть средств, расходуемых в процессе жизненного цикла программного обеспечения.

Такова краткая характеристика *общепринятой модели*. В литературе встречается много вариантов, развивающих ее в сторону детализации и добавления промежуточных фаз, этапов, стадий и отдельных работ (например, по документированию и технической подготовке проектов) в зависимости от особенностей программных проектов или предпочтений разработчиков.

- **Классическая итерационная модель**

Общепринятая модель жизненного цикла не является идеальной уже потому, что только очень простые задачи проходят все этапы без каких-либо *итераций* — возвратов на предыдущие шаги производственного процесса. При программировании, например, может обнаружиться, что *реализация* некоторой функции очень громоздка, неэффективна и вступает в противоречие с требуемой от системы производительностью. В этом случае необходимо перепроектирование, а может быть, и переделка *спецификаций*. При

разработке больших нетрадиционных систем итеративность возникает регулярно на любом этапе жизненного цикла как из-за допущенных на предыдущих шагах ошибок и неточностей, так и из-за изменений внешних требований к условиям эксплуатации системы.

Рис. 3.2. Схема классической итерационной модели жизненного цикла проекта

Классическая итерационная модель абсолютизирует возможность возвратов на предыдущие этапы. Однако это обстоятельство отражает существенный недостаток программных разработок, проводимых в традиционном стиле: стремление заранее предвидеть все ситуации использования системы и невозможность в подавляющем большинстве случаев достичь этого. Все подобные методологии программирования направлены лишь на то, чтобы минимизировать возвраты. Но суть от этого не меняется: при возврате всегда приходится повторять построение того, что уже считалось готовым.

Иначе обстоит дело с методологиями, которые реализуют поддержку итерационного *развития* проектов. В этом случае отказываются от *завершенности фаз и этапов*, вместо чего предлагается распределять наращивание функциональности и интерфейсных возможностей по *итерациям*. В результате можно ослабить требование переделки старого при возвратах. По существу, классическая схема остается верной, но только в рамках одной

итерации и с одной важной поправкой: все полезное, что было сделано ранее, сохраняется.

- **Каскадная модель**

Более строгой разновидностью классической итерационной модели является так называемая ***каскадная модель***, которую можно рассматривать в качестве показательного примера того, какими методами можно минимизировать возвраты.

Характерные черты каскадной модели:

- завершение каждого этапа (они почти те же, что и в классической модели) проверкой полученных результатов, с целью устранить как можно большее количество проблем, связанных с разработкой изделия;
- циклическое повторение пройденных этапов (как в классической модели).

Мотивация *каскадной модели* связана с так называемым *управлением качеством* программного обеспечения. В связи с ней уточняются понятия этапов, некоторые из них структурируются (*спецификация* требований и *реализация*).

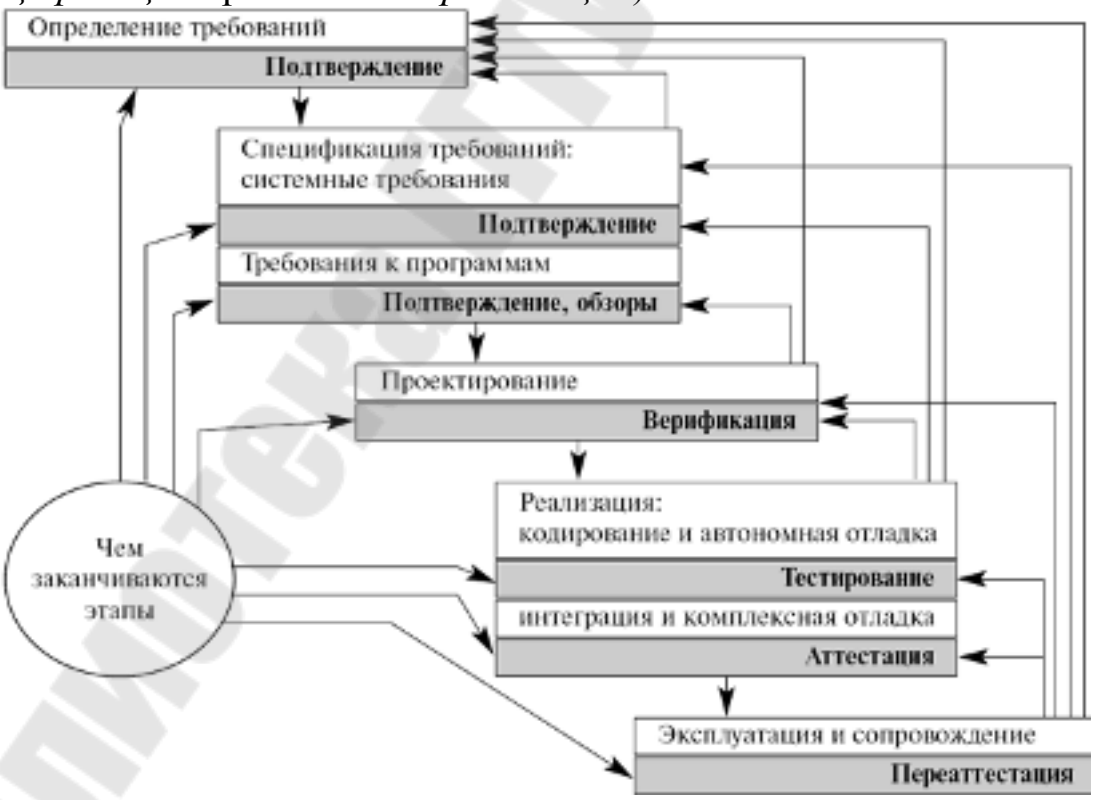


Рис. 3.3. Каскадная модель

Результат *проектирования* верифицируется, т.е. проверяется, обеспечивают ли принятая структура системы и реализационные механизмы выполнимость специфицированных функций.

Реализация контролируется путем *тестирования* компонентов, а после интеграции компонентов в систему и комплексной отладки проводится аттестация, т.е. проверка-фиксация фактически реализованных функций системы, описание ограничений *реализации* и т.п.

В *каскадной модели* верификация и аттестация приписаны к разным этапам, а потому при поверхностном взгляде можно подумать, что это одна и та же деятельность, относящаяся к различным проектным результатам. Однако если рассматривать их как метод проверки каких бы то ни было проектных результатов, то нужно иметь в виду отличие этих родственных видов деятельности. Кратко их можно охарактеризовать следующим образом:

- верификация отвечает на вопрос, правильно ли создана программная система;
- аттестация отвечает на вопрос, правильно ли работает программная система.

Вопросы для самоконтроля:

1. Что такое жизненный цикл проекта?
2. В чем отличие последовательного и итеративного развития проекта?
3. Назовите модели традиционного представления о жизненном цикле проекта.
4. Назовите характерные черты каскадной модели.

4. МОДЕЛИ ФАЗЫ-ФУНКЦИИ, ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА

4.1. Производственные функции в моделировании жизненного цикла: модель фазы—функции

Наиболее последовательно такое дополнение классической схемы реализовано в *модели Гантера* в виде матрицы "*фазы — функции*". Уже из упоминания о матрице следует, что *модель Гантера* имеет два измерения:

- *фазовое*, отражающее *этапы* выполнения проекта и сопутствующие им события;
- *функциональное*, показывающее, какие *производственные функции* выполняются в ходе развития проекта, и какова их интенсивность на каждом из *этапов*.

Фазовое измерение

В *модели Гантера* отражено то, что выполнение *функции* на одном *этапе* может продолжаться на следующем. На рис. 4.1 представлено фазовое измерение модели. Жирной чертой (с разрывом и стрелкой, обозначающей временное направление) изображен процесс разработки. Контрольные точки и наименования событий указаны под этой чертой. Они пронумерованы. Все развитие проекта в модели привязывается к этим контрольным точкам и событиям.

Рис. 4.1. Фазовое измерение модели фазы — функции

В данной *модели* жизненный цикл распадается на следующие перекрывающие друг друга *фазы* (*этапы*):

• ***Этап исследования*** — начинается, когда необходимость *разработки* признана руководством проекта (контрольная точка 0), и заключается в том, что для проекта обосновываются необходимые ресурсы (контрольная точка 1) и формулируются требования к разрабатываемому изделию (контрольная точка 2).

• ***Анализ осуществимости*** — начинается на *этапе исследования*, когда определены исполнители проекта (контрольная точка 1), и завершается утверждением требований (контрольная точка 3). Цель *этапа* — определить возможность *конструирования* изделия с технической точки зрения (достаточно ли ресурсов, квалификации и т.п.), будет ли изделие удобно для практического *использования*; решение вопросов экономической и коммерческой эффективности.

• ***Конструирование*** — начинается обычно на *этапе анализа осуществимости*, как только документально зафиксированы предварительные цели проекта (контрольная точка 2), и заканчивается утверждением проектных решений в виде официальной спецификации на *разработку* (контрольная точка 5).

• ***Программирование*** — начинается на *этапе конструирования*, когда становятся доступными основные спецификации на отдельные компоненты изделия (контрольная точка 4), но не ранее утверждения соглашения о требованиях (контрольная точка 3). Совмещение данной *фазы* с заключительным *этапом конструирования* обеспечивает оперативную проверку проектных решений и некоторых ключевых вопросов *разработки*. Цель *этапа* — реализация программ компонентов с последующей сборкой изделия. Он завершается, когда разработчики заканчивают документирование, отладку и компоновку и передают изделие службе, выполняющей независимую *оценку* результатов работы (независимые *испытания* начались — контрольная точка 7).

• ***Оценка*** — является буферной зоной между началом *испытаний* и практическим *использованием* изделия. *Этап* начинается, как только проведены внутренние (силами разработчиков) *испытания* изделия (контрольная точка 6) и заканчивается, когда подтверждается готовность изделия к эксплуатации (контрольная точка 9).

• ***Использование*** — начинается ближе к концу *этапа оценки*, когда готовность изделия к эксплуатации проверена и может организовываться передача изделия на распространение (контрольная точка 8). *Этап* продолжается, пока изделие находится в действии и интенсивно эксплуатируется. Он связан с внедрением, обучением, настройкой и *сопровождением*, возможно, с модернизацией изделия. *Этап* заканчивается, когда разработчики прекращают систематическую деятельность по *сопровождению* и *поддержке* данного программного изделия (контрольная точка 10).

Функциональное измерение

На протяжении *фаз* жизненного цикла разработчики выполняют различные организационные и *производственные функции*, которые естественным образом группируются в классы родственных *функций*. Некоторые из этих *функций* для того или иного сотрудника являются технологическими, т.е. имеют четкий регламент выполнения (см. лекцию 1), другие, напротив, требуют разъяснения для выполнения. Здесь это не принципиально, но с точки зрения моделирования жизненного цикла важно следующее:

• классы родственных *функций* можно считать выполняемыми в течение всего хода развития проекта;

• содержание (цели) *функции* на различных *этапах* претерпевает изменение;

• интенсивность *функции* (в частности, потребности в ресурсах, необходимых для выполнения) меняется как при переходе от *этапа* к *этапу*, так и в рамках работ одного *этапа*.

Исходя из этих предпосылок, Гантер строит второе измерение своей модели. Он говорит о следующем наборе *функций* (если угодно — классов *функций*).

• ***Планирование*** — *функция*, которая должна выполняться с самого начала и до конца развития любого проекта. О содержании того, что должно планироваться на каждом из *этапов*, можно судить по наименованиям контрольных точек. Конкретные методы *планирования* определяются концепциями, принимаемыми для данного проекта. Различаются стратегическое и текущее *планирование*, и оба вида непосредственно относятся к деятельности менеджера проекта.

• ***Разработка.*** Как и *планирование*, эта *функция* пронизывает весь проект. Содержание ее, т.е. то, что нужно разрабатывать, меняется, но в целом его можно охарактеризовать как получение рабочего продукта *этапа*. По-видимому, для *оценки разработки* показателей интенсивности недостаточно, и нужно привлекать сведения о соответствии целям (связь с *планированием*), о расходовании ресурсов и др.

• ***Обслуживание*** — *функция*, обеспечивающая максимально комфортную обстановку выполнения некоторой деятельности. Обслуживаемые виды деятельности меняются при переходе от одного *этапа* к другому, а могут распространяться и на несколько *этапов*. Таким образом, по интенсивности *обслуживания* можно косвенно судить о качестве организации работ проекта. Особенностью любого *обслуживания* является то, что в нем участвуют как минимум два субъекта: обслуживаемый и обслуживающий. Поэтому говорить об *обслуживании* до того момента, когда в проекте начнут действовать исполнители, не приходится (иными словами, должно быть если не принято, то хотя бы обосновано намерение выполнять проект).

• ***Выпуск документации.*** Документация в проекте включает в себя не только техническое описание системы. Она рассматривается как полноправный вид рабочих продуктов, сопровождающий другие рабочие продукты: программы, диаграммы моделей системы и прочее. Первый рабочий продукт в проекте — утвержденные

требования. И именно он первым оформляется как документ. Поэтому до утверждения требований говорить о *выпуске документации* как о выполняемой *производственной функции* не приходится. Вместе с тем начиная с этого момента интенсивность деятельности по *выпуску документации* растет и достигает своего максимума на *этапе программирования* перед *этапом оценки*.

• ***Испытания***. Испытывать приходится все рабочие продукты проекта. О вариантности содержания этой *функции* можно судить по целям испытательных работ: подтверждения, обзоры, верификация, тестирование, аттестация и переаттестация (см. обсуждение каскадной модели в лекции 7). Поскольку испытывать можно лишь то, что готово для *испытания*, активизация *функции* начинается тогда, когда первичные требования к проекту сформулированы и нуждаются в проверке.

• ***Поддержка*** *использования* рабочих продуктов — это *функция*, выполнение которой необходимо в связи с передачей продукта в эксплуатацию. Содержание ее связано с обучением и созданием необходимой для *использования* продуктов инфраструктуры. Пользователями результатов анализа сначала являются разработчики архитектуры, а затем другие разработчики. Пользователи результатов *конструирования* — программисты, а затем другие заинтересованные лица. По этой схеме организуется эксплуатация всех рабочих продуктов, в том числе и поставляемых программных продуктов. Всякий раз требуется свой содержательный уровень *поддержки*.

• ***Сопровождение*** по Гантеру отличается от *поддержки* тем, что оно организуется для внешнего *использования* продуктов. Оно предполагает организацию *поддержки* и на этой основе выстраивает мероприятия, нацеленные на активную обратную связь с пользователями, чтобы можно было реагировать на их отклики, в том числе на изменение требований к продукту. Другой мотивировкой выделения *сопровождения* как особой *функции* является то, что для этих видов работ чаще всего приходится выделять специальные кадровые ресурсы, в частности со своей ролевой структурой, которая не сводится к структуре проектной группы.

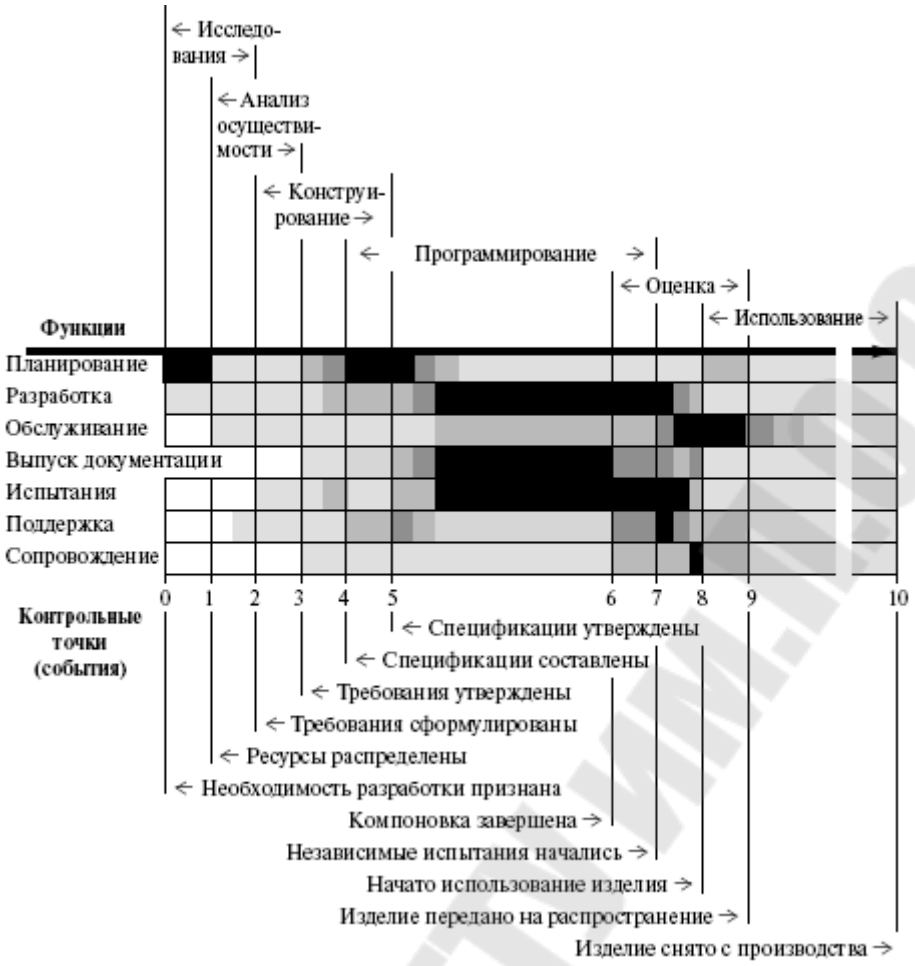


Рис. 4.2. Матрица фазы—функции модели Гантера

Для модели особенно важно, что приведенные и, возможно, другие *функции* **совмещаются при реализации проекта**, на базе чего строится функциональное измерение модели, наложение которого на фазовое измерение дает изображение *матрицы фаз–функций* в целом (см. рис. 4.2, на котором интенсивность выполняемых *функций* отражается густотой закраски клеток матрицы).

Учет итерационного развития

Модель фазы–функции не описывает возможности итеративного возврата на предыдущие *этапы*. Строгую каскадность в ней еще можно усмотреть в изображении перекрытия *этапов*. Но для отражения произвольных возвратов модель в том виде, в котором она предлагается Гантером, непригодна. Тем не менее можно предложить один прием, с помощью которого этот недостаток легко ликвидировать.

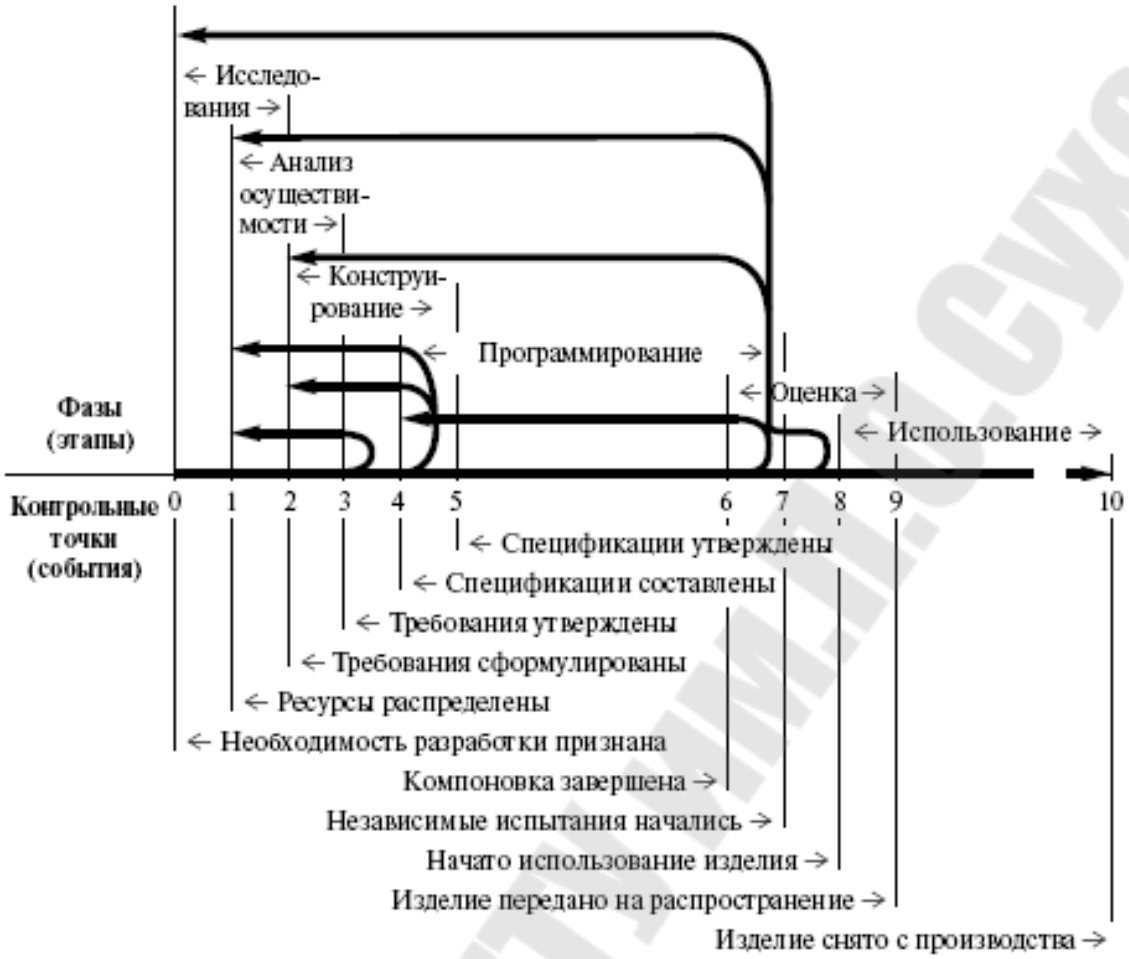


Рис. 4.3. Учет итеративности в модели фазы–функции

Расщепление линии жизненного цикла может означать два варианта хода развития проекта.

• Приостановка основного процесса, переход к точке продолжения, в которой возобновляются указанные на схеме действия, выполнение процесса до точки *расщепления* и слияние линий жизненного цикла, которое понимается как возобновление основного процесса. Эта схема допускает, что приостановка может осуществляться в любой момент — нужно только позаботиться о мерах, которые в будущем обеспечат корректное слияние.

• *Действительное расщепление*, при котором основной процесс не прекращается и развивается в соответствии со своей линией, а переход к точке продолжения рассматривается как организация дополнительного процесса, вкладывающегося в жизненный цикл. Ресурсы на такое *расщепление*, а также механизмы взаимосвязей двух линий должны предусматриваться заранее. В этом случае возврат можно считать не вынужденной мерой, а плановым переходом, отражающим ревизию сделанного. Однако плановость предполагает

определенный регламент поведения и, в частности, то, что *расщепление* разрешается не произвольно, а в конкретные моменты развития жизненного цикла, которые естественно рассматривать в качестве дополнительных контрольных точек.

4.2. События в объектно-ориентированном проектировании

Принципиальные моменты, в которых *объектно-ориентированный подход* к развитию проектов стоит сопоставить с традиционными последовательными методологиями, сводятся к следующему:

1. Итеративность развития.
2. Изменение функциональности.
3. Формирование системы понятий проекта.
4. Наращивание функциональности в соответствии со сценариями.
5. Ничто не делается однократно.
6. Оперирование на размножающихся фазах подобно.
7. Распределение реализуемых требований по итерациям.
8. Особый стиль наращивания возможностей системы и ее развития.

Рис 4.4. События в объектно-ориентированном проектировании

По вполне понятным причинам в объектно-ориентированном проектировании несколько изменяется содержание ряда этапов, что нашло отражение в количестве и наименованиях событий на рис. 4.4.

Веха 0. Необходимость разработки признана.

Эта точка означает начало работ над проектом, которые еще не утверждены как официальный проект. Официальным он становится,

когда доказана его осуществимость, задачи проекта в целом обозначены, в том числе определены те задачи, которые предстоит решать в первую очередь (*контрольная точка* 2). В этот период можно говорить лишь о предпроектной деятельности менеджера. Если схема развития проекта допускает распределение менеджерских обязанностей в команде исполнителей и есть команда, которая принимает решение о целесообразности работ над проектом, предпроектная деятельность менеджера становится сферой ответственности команды целиком.

Веха 1. Ресурсы распределены.

Для начала работ над проектом необходимо знать, какими *ресурсами* можно располагать. В данной точке фиксируется, что предполагает планировщик предоставить менеджеру, оцениваются средства, которые разумно ожидать от заказчика и от инвесторов. Эти сведения есть результат соответствующей деятельности менеджера. В частности, он определяет основы концепции развития проекта, мотивирующие потребность.

Веха 2. Требования к очередной итерации сформулированы, общие требования и общий план составлены, ближайшая задача, критерии оценки результатов и перспективные задачи определены.

Эта *контрольная точка* обозначает начало **стационарного цикла развития проекта**, когда исходя из имеющейся информации планируется ход текущей итерации. Для первой итерации эта информация собирается на **начальной фазе проекта**, которая в данный момент завершается. Исходная информация для последующих итераций извлекается из всей предшествующей истории проекта, она формируется в доступном для работы виде в ходе этапа оценочных работ предыдущей итерации к моменту *расщепления жизненного цикла*.

Для ближайшей задачи должны быть определены:

- планы реализации — разбиение решения на этапы, сроки их выполнения и ресурсные потребности, определяемые для реализации конкретных требований ближайшей задачи;
- ***критерии оценки результатов*** — методики, с помощью которых определяется, что конкретные требования ближайшей задачи реализованы в срок с необходимым качеством;
- ***перспективные задачи***, формулировка которых в проектах жесткой отчетности является обязательным результатом,

проверяемым в контрольной точке 2. Их предполагается решить на последующих итерациях (степень проработки требований к *перспективным задачам* и даже их идентификация могут быть различны).

Последнее не требуется, например, в схемах экстремального программирования, однако и для подхода быстрого развития полезно выработать гипотезы о перспективах проекта, которые нужно будет иметь в виду на следующих этапах (итерациях).

В контрольной точке 2 определяется фронт работ проектировщиков подсистем.

Веха 3. Требования к очередной итерации утверждены.

Все сведения о проекте, представленные в контрольной точке 2, к моменту прихода к третьей контрольной точке должны быть согласованы для утверждения. Для больших и сложных проектов данный момент определяется формально, для него заданы отчетные материалы, которые утверждаются. В более простых случаях этого не требуется, тем не менее веха, знаменующая окончание аналитического этапа проекта (итерации), существует независимо от формальной стороны дела. Ее отслеживание просто необходимо менеджеру в качестве момента, когда подводятся первые итоги проекта (итерации).

Веха 4. Спецификации реализуемых сценариев составлены.

Начало этапа конструирования связывается с декомпозицией решаемых проектом (итерацией) задач и с построением архитектуры системы.

Коль скоро архитектура определена, пусть даже лишь в общих чертах, появляется фронт работ для разработчиков подсистем. Соответственно, у руководителей команд разработчиков появляется сфера ответственности для текущей итерации. Менеджерские обязанности в проекте, которые становятся главными в этой точке, — оформление подготовленных для реализации *сценариев* к утверждению. Как и в предыдущем случае, данная *контрольная точка* вполне может быть явно не выделена, но это не означает, что работа над трансформацией *сценариев* в архитектуру растворяется в проекте.

Веха 5. Спецификации утверждены.

Эта *контрольная точка* обозначает окончание этапа конструирования. Архитектура для очередной итерации утверждена и зафиксирована в виде заданий для разработчиков подсистем и их

руководителей, от которых требуется создание или модернизация наборов классов проекта, относящихся к их сферам ответственности.

Веха 6. Автономная проверка завершена и комплексное тестирование началось.

По мере продвижения этапа программирования к завершению возникает момент, когда требуется комплексная проверка работоспособности системы. Он означает начало этапа оценки, поскольку с этого момента появляется возможность проверить предварительные суждения о проекте (итерации) на практике.

Если иметь в виду мероприятия, которые предполагается осуществлять при прохождении указанных точек, то методы быстрого развития действительно их игнорируют. Однако соответствующие процессы, о которых только что шла речь, отменить не удается. Разработчики в любом случае вынуждены заботиться и о точном специфицировании того, что требуется реализовать, и об архитектуре системы в целом. По существу, экономия времени достигается, когда слияние контрольных точек не противоречит, например, сложности проекта, перспективам его развития и т.д., и именно за счет того, что ликвидируется сдерживающий регламент развития системы. К сожалению, это далеко не всегда оправданно.

Веха 7. Тестирование завершилось, начата подготовка новой итерации.

Эта *контрольная точка* отмечает окончание этапа программистских работ, определенных текущей итерацией. Программистские работы продолжаются как деятельность, связанная с *пополнением базового окружения проекта*, которое мы выделяем как относительно самостоятельный этап проекта, вложенный в этап оценки, и обычно завершающийся вместе с ним (*контрольная точка* 10). Эти работы подразделяются на два вида:

- выделение общих лишь для данного проекта переиспользуемых компонентов (его целесообразно выполнить до *расщепления жизненного цикла — контрольная точка* 9);
- выделение общих, не привязанных к проекту переиспользуемых компонентов (целесообразно начинать эти работы, когда программный рабочий продукт итерации рассматривается как **готовое приложение**, — *контрольная точка* 9, и завершать к моменту передачи системы на распространение — *контрольная точка* 10).

Итеративное развитие проекта обеспечивается сбором сведений для новой итерации, которая начинается в рассматриваемой контрольной точке.

Веха 8. Требования к новой итерации приняты.

В ходе обработки сведений для новой итерации возникает момент, когда их можно оформить как принимаемые требования. Набор таких требований пополняется в течение первого периода использования силами разработчиков (этот период обычно называют альфа-тестированием). По его завершении происходит *расщепление жизненного цикла* с целью параллельного выполнения передачи системы в эксплуатацию и организации работ новой итерации.

Расщепление приводит к одновременному существованию двух версий системы: одна из них начинает использоваться (отменяет ли она другие, ранее существовавшие версии — вопрос специфики проекта), а вторая зарождается в виде набора требований. Это время, когда должна быть произведена ревизия априорных гипотез, предположений, корректировка показателей и нормативов проекта.

Веха 9. Начато использование изделия.

Эта *контрольная точка* обозначает начало так называемого бета-тестирования, т.е. передачу системы в эксплуатацию для внешнего оценивания. После выполнения соответствующих работ, и в частности исправления обнаруженных дефектов, можно переходить к распространению результатов итерации (*контрольная точка* 10).

Контрольная точка 9 отмечает начало так называемой **фазы завершения проекта (итерации)**. Она охватывает часть жизненного цикла, которая отражает деятельность разработчиков, связанную рабочими продуктами итерации после получения результатов. Фаза завершения подобна традиционной фазе эксплуатации и сопровождения, однако есть и отличия, обусловленные тем, что объектно-ориентированный проект обычно имеет дело с иерархиями версий системы, отражающими наращивание возможностей. Данная фаза пересекается с этапом оценки.

Традиционные работы фазы завершения включают в себя:

- *поставку или пакетирование изделия* для потребителя (*контрольная точка* 9);
- сопровождение программного продукта (по причине разнообразия вариантов организации этих работ они редко описываются структурно, т.е. с разбиением на этапы);
- этап окончания работ (см. ниже — контрольные точки 11, 12).

Веха 10. Изделие или его версия переданы на распространение.

Этап использования в данной контрольной точке приобретает новое качество: у версии появляется круг пользователей, нуждающихся в обслуживании.

Веха 11. Извещение о прекращении поддержки изделия (версии) выпущено.

Начало этапа окончания работ: оповещение о прекращении сопровождения и сворачивание деятельности по поддержке версии или всех версий к определенному сроку.

Предварительное оповещение о наступлении этапа необходимо пользователям, чтобы они смогли либо перестроиться, либо найти аргументы (в частности, ресурсы) в пользу продолжения сопровождения изделия.

Веха 12. Изделие (версия) снято с производства.

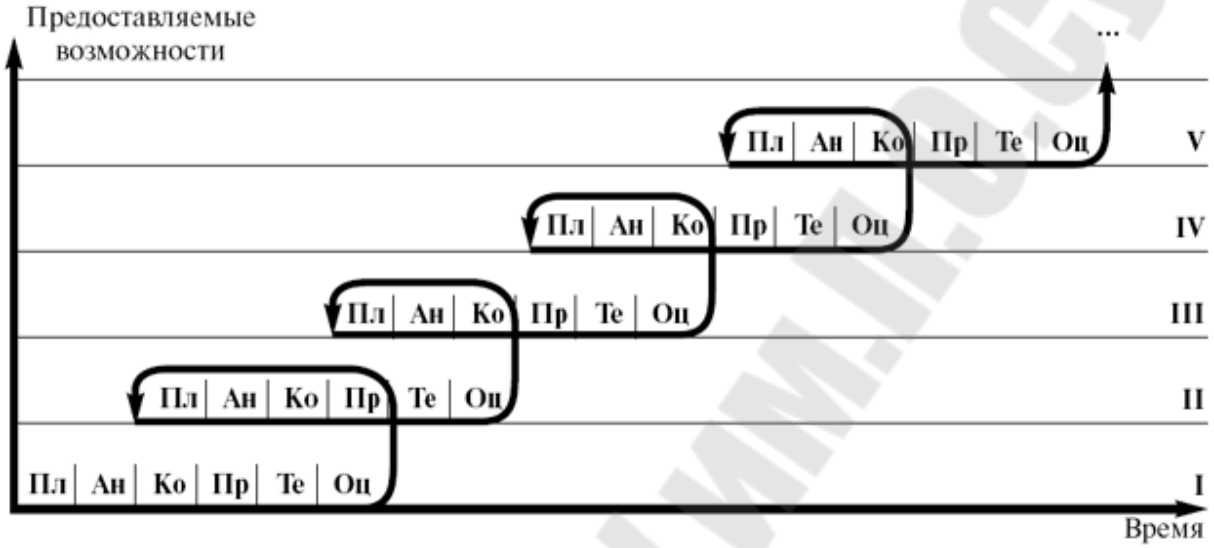
Завершение этапа окончания работ: прекращение сопровождения и сворачивание деятельности по поддержке версии или всех версий. После получения извещения о прекращении поддержки изделия (версии) некоторые пользователи, возможно, захотят отложить (на определенный или неопределенный срок) наступление данного события. Если они в состоянии финансировать поддержку, сопровождение и, возможно, развитие изделия, то срок окончания работ передвигается. Это обычная процедура, благодаря которой продолжают жить и даже совершенствоваться устаревающие системы. Причины откладывания прекращения поддержки вполне понятны: это стремление не потерять своих пользователей пусть даже ценой существенных дополнительных затрат. Классический пример тому — уникальное долгожительство языка Fortran.

Этап окончания работ мог бы быть представлен во всех традиционных моделях, но в то время, когда эти модели разрабатывались, ему не придавали особого значения. Вместе с тем, когда речь идет о совместной поддержке нескольких версий (а именно такая ситуация типична для объектно-ориентированного проектирования), *окончание работ* игнорировать нельзя.

4.3. Спираль развития

В модифицированной модели Гантера не был наглядно выделен аспект итеративного подхода, связанный с постепенным наращиванием возможностей системы по мере перехода от одной

итерации к другой. Мы абстрагировались от него, делая акцент на обсуждении этапов и *производственных функций*. В результате была получена схема, на которой линия жизни и функциональное измерение оказывались сбалансированными настолько, что модель вполне могла бы служить основой для разработки соответствующего CASE-средства.



Спираль охвата предметной области

Постепенное наращивание возможностей системы по мере развития проекта часто изображают в виде спирали, раскручивающейся от центра, как показано на рис. 10.3. В соответствии с этой простой (грубой) моделью развитие проекта описывается как постепенный охват все более расширяющейся области плоскости по мере перехода проекта от этапа к этапу и от итерации к итерации. Данная модель делает акцент на том, что объектно-ориентированное развитие приводит к постепенному расширению сферы применения конструируемой системы.

Про объектно-ориентированное развитие часто говорят, что традиционные этапы жизненного цикла разработки никогда не кончаются. *Спираль охвата* наглядно иллюстрирует смысл этого тезиса.

В данной модели можно усмотреть еще один аспект конструирования программных систем — типичную схему развития коллектива разработчиков, который начиная со своего первого проекта постепенно пополняет накапливаемый багаж переиспользуемых рабочих продуктов и, что, пожалуй, еще важнее, — опыт работы и квалификацию.

Модель расширения охвата прикладной области совсем не претендует на *инструментальность*. Поэтому обсуждать эти свойства для данной модели мы не будем.

Вопросы для самоконтроля:

1. Назовите фазы в модели фаз-функций.
2. Назовите функции в модели фаз-функций.
3. Как можно учесть итерационное развитие проекта в модели фаз-функций?
4. Назовите события (вехи) в модели объектно-ориентированного проектирования.
5. Что такое спираль развития проекта. Назовите основные сектора спирали развития?

5. УПРАВЛЕНИЕ РИСКАМИ И КАЧЕСТВОМ ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ ПРОЕКТОВ

5.1. Управление рисками

- ***Причины риска*** — определенные события или обстоятельства в проекте или в его окружении, которые вызывают неопределенность.
- ***Риски*** — неопределенные события или обстоятельства, возникновение которых может привести к воздействию на процесс достижения целей проекта:
  - к негативному воздействию, если в результате траектория проектной деятельности выходит из области допустимости;
  - нейтральному воздействию, если траектория не выходит из этой области;
  - к позитивному воздействию, когда новая траектория не только остается допустимой, но и по разным причинам оказывается предпочтительнее других траекторий, которые могли бы реализоваться, если бы *рисковая ситуация* не возникла;
- ***Последствия*** — незапланированные отклонения траектории выполнения проекта, возникшие в результате того, что событие или обстоятельство, квалифицированное как риск, имели место.

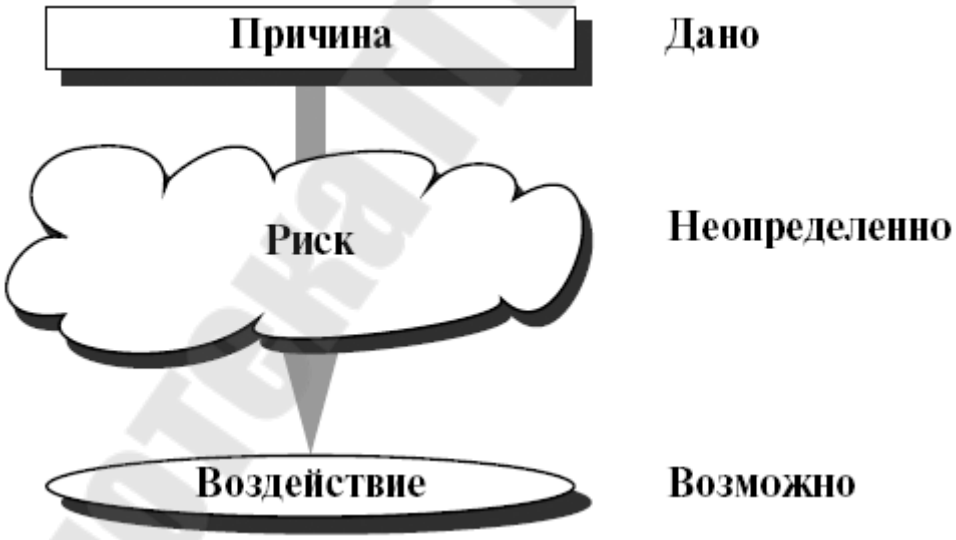


Рис. 5.1. Составляющие риска

Реальные риски — это те события, которые действительно характеризуются неопределенностью. Если причина детерминированно ведет к событию, то это не риск, а штатная ситуация, план преодоления которой к *управлению рисками* отношения не имеет. Следовательно, нужен анализ связей причин и событий.

В результате идентификации рисков формируется список идентифицированных реальных рисков.

Вторая стадия составления плана *управления рисками* при любой методологии — **выставление** приоритетов рискам. Иными словами, нужно определить сравнительную важность рисков для проекта. Устанавливается, насколько существенным может оказаться *воздействие каждого риска на проект*, и по этому признаку все идентифицированные риски упорядочиваются. После этого оцениваются две вероятности: для причины и возможного воздействия. Удобно расположить все риски в виде точек на плоскости, координаты которых отражают заданные вероятности. В таком случае можно зрительно отделить несущественные риски от существенных и далее работать только с последними. Нужно, однако, предостеречь, что выставление вероятностей — операция субъективная, а потому результирующее отсеивание требует проверки.

На третьей стадии продолжается работа с полученным списком. Устанавливается возможность *влияния на риски*, т.е. на причины рисков и на оказываемое ими *воздействие на проект*. Влияние может осуществляться на нескольких уровнях и для каждого из рисков должны быть определены влияния, планируемые для последовательного выполнения на тех уровнях, на которых это возможно:

1. *Исключение риска* (avoid). Некоторые рискованные ситуации можно исключить полностью. Например, чтобы увольнение работника с ключевой ролью не сильно сказалось на продолжении развития проекта, целесообразно с самого начала предложить для исполнения этой роли двух человек сравнимой квалификации. В начале проекта их дискуссии полезны для выработки объективных решений, а если один из них откажется от контракта, второй все еще сможет продолжать дело. Полезные дискуссии — эта та жертва, которая во многих случаях возможна для *исключения риска*. К сожалению, дублирование не может быть рекомендовано для *исключения* всех рискованных ситуаций.

2. *Переключение риска* (transfer). Это частный случай *исключения*, когда риск переносится из сферы влияния проекта на окружение. Например, если рыночная ценность создаваемого изделия кажется сомнительной, для его разработки комфортнее договориться с заказчиком об авансовых платежах, тем самым заставив его самого

решать задачу преодоления риска и освободив от этого разработчиков (возможно, за счет снижения оплаты проекта). К этому уровню относятся все варианты контрактных соглашений, но не только они. Оперируя рисками, всегда нужно стараться предусмотреть способы *переключения*.

3. *Уменьшение риска* (mitigate). Если риск не может быть исключен, можно постараться уменьшить вероятность его появления на практике (оперирование причинами). Продолжая пример с увольнением сотрудника, для снижения вероятности этого следует предугадать причины поступка и постараться создать для сотрудника более комфортные условия (повысить заработную плату, создать льготы и т.п.). Нужно, чтобы подобные решения делались не в ответ на заявление об увольнении, а заранее. Это сохранит определенную стабильность в коллективе, которая сама по себе является методом *уменьшения риска*.

4. *Предупреждение ущерба от риска* (accept). Когда не получается удовлетворительно уменьшить риск, следует подготовиться к встрече с ним так, чтобы минимизировать потери, т.е. снизить вероятность негативного воздействия и уменьшить само воздействие (оперирование последствиями). Если это удается, то продолжение проекта во многих случаях оказывается успешным. В примере с увольнением следует как можно скорее найти замену данному сотруднику. Естественно, время выполнения проекта увеличится (в частности, потому что нового человека придется вводить в курс дела), но работа все-таки будет продолжена. Это так, но при одном условии: на всех уровнях проектирования заложена возможность отделения результатов труда от разработчиков. Если результаты персонифицированы, то трудности подмены для некоторых ролей могут оказаться непреодолимыми.

5.2. Управление качеством проекта

Планирование мероприятий и стратегии *управления качеством* разрабатываемого программного изделия — традиционная часть подготовительных работ, выполняемых менеджером. Они связаны с определением принципов, которые помогают в решении вопросов *качества* разработки, непременно возникающих на всех стадиях развития проекта:

- утверждение целей проекта с точки зрения *качества* его выполнения, определение критериев успеха и достижения ожидаемых результатов;
- проверка и отслеживание *качества*;
- удаление дефектов;
- указание общих показателей *качества*.

План управления качеством — это перечень мероприятий, которые проводятся в контрольных точках жизненного цикла проекта для измерения и оценки определенных показателей, характеризующих достигаемые результаты. Обычно такой план включает:

- цели, достижение которых удовлетворяет заказчика проекта в целом (удобство использования, возможности и эксплуатационные характеристики и др.);
- цели, связанные *качеством* программирования (удобная для чтения структура, комментарии и их полнота, соответствие спецификациям, переиспользуемость и эффективность кода);
- использование инструментов (какие, для чего, как влияют на *качество*);
- модель исправления дефектов (организационная процедура поиска дефектов и их устранения);
- средства *управления качеством*:
  - соглашение об описании процессов в виде, удобном для изменения и исправления программ;
  - представление порядка отслеживания *качества*;
  - специфичные для проекта соглашения.

В приведенном ниже перечне указаны некоторые довольно типичные ошибки проектирования, относящиеся к *качеству*:

- **Подмена автоматизации деятельности предоставлением средств**. Эта очень распространенная ошибка разработчиков возникает в связи с тем, что при определении требований к программному продукту фиксируется, что нужно предоставить пользователю, но игнорируется вопрос о том, как данные средства будут включены в его деятельность. В результате выполнение типичных и частых для данной деятельности действий требует усилий со стороны пользователя, что снижает эффективность средства.
- **Игнорирование средств окружения, которые можно использовать для автоматизируемой деятельности**. В ряде случаев

средства, предоставляемые окружением программы, дублируются. Примеры такого рода: различные мини-редакторы, визуализаторы, копировщики. Иногда это оправданно, но чаще дублирующие средства реализуются только по той причине, что они нужны для данной деятельности. В результате повышается стоимость разработки, а пользователю приходится осваивать дополнительные средства. Как и в предыдущем случае, для исключения дублирования необходимо специальное исследование для определения возможности и целесообразности использования существующих средств (аспект общего плана проекта), а также проверяемые показатели переиспользования (аспект *плана управления качеством*).

- **Пользовательский интерфейс, не соответствующий автоматизируемой деятельности**: непривычный, эргономически неточный и т.д. Причина этого недостатка все та же: просчеты с исследованием автоматизируемой деятельности. Но кроме того — игнорирование в общем плане проекта задачи специальной настройки интерфейса.
- **Неадекватная реакция системы на ошибки пользователя**. Правильная точка зрения на пользовательские ошибки — считать их не исключениями, а обычными явлениями, возникающими при работе программы. В этом случае можно добиться в диагностике ответа на три вопроса, возникающие у пользователя, когда система фиксирует ошибку:
  - **Что случилось?** В диагностическом сообщении причина ошибки и ее последствия должны разъясняться в пользовательских терминах.
  - **Что можно сделать?** Пользователю должна быть предоставлена возможность получить информацию о возможных действиях, на корректное выполнение которых он может рассчитывать.
  - **Как исправить положение?** Следует дать пользователю возможность получить сведения о пути исправления ошибки с уточнением, какие потери его ожидают.

Систему, диагностика которой удовлетворяет указанному критерию во всех ошибочных случаях, следует признать более качественной. К сожалению, и здесь трудно определить количественные показатели *качества*.

Подобный критерий можно определить для предупреждающих сообщений.

- **Неадекватная реакция на системные ошибки (ошибки разработчиков)**.

Любая программная система не застрахована от ошибок. В качественной системе, заботящейся о комфортных условиях работы пользователя, предусматриваются рекомендации действий, когда обнаруживается ошибка разработчиков. Кроме того, пользователь должен быть информирован о процедуре внесения исправления. Понятно, что для обеспечения этого требуются специальные ресурсы.

Вопросы для самоконтроля:

1. Какие существуют способы уменьшения и исключения рисков?
2. Что включает в себя план управления качеством проекта?
3. Назовите типичные ошибки проектирования, относящиеся к качеству.

6. ПРОБЛЕМЫ И ОШИБКИ ПРИ РАЗРАБОТКЕ ВЕБ-ПРОЕКТОВ

6.1. Распространенные проблемы при управлении веб-проектами

- **Общее отношение:**

Большинство менеджеров веб-проектов **вырастает из разработчиков**.

При этом часто возникает ряд проблем.

- Не у всякого исполнителя есть нацеленность на результат. Порой, работа воспринимается как процесс. Тогда как, у менеджера, должна присутствовать личная заинтересованность в своевременной сдаче проекта. И это должно заключаться не столько в получении процентов, сколько в определенном настрое, мировоззрении.
- Чрезмерный интерес к реализации. То есть вместо управленческих функций, менеджер начинает диктовать правила написания кода или формирования архитектуры.
- Можно привести в пример реальный случай, когда разработчика повысили до ПМ. Через некоторое время, он заглянул в код, управляемого им проекта, и ужаснулся. “Если хочешь, чтобы что-то было сделано хорошо – сделай это сам” - сказал он и взялся за кодирование. При этом он был и остается отличным специалистом. Возможно, это была его личная неудача, так как когда ПМ начинает заниматься кодом, то отвлекается и теряет контроль над проектом.
- Характер человека так же играет немаловажную роль. Довольно часто можно встретить ситуацию, когда проблемы возникают из-за не сделанного в нужное время телефонного звонка или из-за неотправленного письма.
- Стоит отметить и общее отношение к ошибкам. Бывает и так, что из-за страха перед начальством некоторые проблемы остаются не озвученными. Что через некоторое время может привести к более серьезным последствиям, чем, если бы проблема была решена на ранней стадии.
- И самый тяжелый случай: “Менеджеры меня гоняли, теперь уж погоняю я”.

То есть речь не идет об управлении, а скорее о самоутверждении. При этом забывается, что основная задача менеджера это не гонять сотрудников, а создать людям условия для

работы и контролировать процесс (чтобы в заданные сроки, за заданный бюджет была реализована данная функциональность). Недостаточный контроль – так же частая проблема.

- Очень тяжело бывает работать с руководителями, которые считают, что люди – это ресурс. И это притом, что и менеджеры, и разработчики все те же наемные работники, просто у них различные роли в проекте.

- **Сроки**

Наверное, одна из самых больных тем. Рынок веб-девелопмента насыщен. И за каждого клиента приходится биться, придумывая и сроки поменьше и цены пониже. Это сильно сказывается на состоянии команды разработчиков, которым часто приходится работать в состоянии аврала.

Иногда встречается и другой подход. Сроки выполнения задач уточняются у исполнителей. Однако об адекватности подобных оценок мало кто задумывается.

Бывает и так, что сроки навязываются заказчиком. Это не такой уж и плохой вариант как кажется. Но только при том условии, если заказчик готов согласовать объем работ и принять бюджет (или же наоборот).

- **Отсутствие плана проекта**

Конечно же, план есть. Только в голове менеджера и не документирован. Хотя, обычно это совсем не план, а общее понимание необходимых работ для завершения проекта. В управлении проектами основная задача – это создание предсказуемости. Что к данному моменту времени можно с высокой вероятностью назвать выполненный (и не выполненный) объем работ. В случае экстренных ситуаций план может помочь определить текущее состояние проекта, быстрее сориентироваться, и принять верное решение. Наличие плана дает еще возможность контроля над ходом работ.

- **Бессистемность при управление требованиями**

Любой проект должен начинаться со сбора и анализа требований. Как иначе понять, что действительно нужно клиенту? Проблема в том, что не всякий заказчик четко понимает, что именно он хочет от данного проекта. При создании сайтов это довольно распространенная ситуация. В подобной ситуации есть два основных решения.

Первое – провести переговоры, и определить реальные требования и желаемый результат от создания проекта. Да, это не всегда просто. Порой одних переговоров может быть не достаточно. Однако с подобным подходом повышается вероятность, что заказчик получит действительно то, на что он хотел потратить свои деньги.

Второе решение более популярно. Основано оно на установке “Мы знаем, что Вам нужно”. При выборе данного подхода продается продукт, составленный из готовых модулей. Для веб – разработок, пожалуй, это довольно важно. Действительно, регулярное создание сайтов “с нуля” (если такое вообще возможно при большом потоке заказов) – это излишнее расточительство ресурсов. Но в таком случае, заказчику навязывается ряд ограничений, имеющихся у данного готового продукта. Соответственно чаще возникает ситуация, когда проект готов, а заказчик говорит: “Но получилось совсем не то, что мы хотели”.

Бессистемность сбора требований ведет к следующим проблемам:

- Объем работ растет, причем не контролируемо.
- Сроки изменяются произвольно
- Невозможно провести полноценное проектирование при нечетких требованиях. Это может отразиться на качестве конечного продукта
- В любой момент времени заказчик может отказаться от своих слов. Как итог - потерянное время (в лучшем случае)
- Общий список требований не согласовывается. Соответственно, конечный продукт может не удовлетворять заказчика

- **Отсутствие управления рисками**

В веб-проектах, как правило, об управлении рисками никто не думает. Понятно, что они, в общем-то, одни и те же: неадекватная оценка объема работ и сроков, задержки при согласовании ТЗ или макетов, несвоевременное предоставление оборудования и так далее. Эти риски игнорируются, хотя правильнее было все-таки указать в договоре о выполнении работ варианты действий на в этих случаях. А регулярно сорванные сроки – это риск испортить репутацию.

- **Отсутствие совещаний**

Проведение совещаний по веб-проекту – довольно большая редкость. Во всяком случае, если они и проводятся, то подготовить и заблаговременно разослать повестку обычно забывают. Естественно,

что с таким подходом, про составление протокола вообще речи не идет. Сами же совещания частенько сводятся к парным обсуждениям технических вопросов. В итоге, получается пустая трата времени: что обсуждали – не понятно, какие решения приняты – никто не запомнил, что дальше делать – тоже не ясно.

- **Работа над ошибками**

Менеджеры веб-проектов обычно даже не запоминают, сколько проектов у них было. Получается, что опыт ведения проектов не накапливается. Так как при отсутствии привычки составлять соответствующие документы, с менеджерами из компании будут уходить и их знания, которыми не всегда можно поделиться при сдаче-приеме дел. Соответственно, процесс развития в управлении может двигаться крайне медленно.

Подведем итог

Возможно, основная проблема веб-девелопмента в незрелости. В этой сфере очень много молодых специалистов без опыта или специального образования. Многие, получив этот самый начальный опыт, переходят к более серьезным софтверным проектам, освобождая место новичкам.

Иногда возникает вопрос, а **целесообразно ли усложнять процесс разработки сайтов,** внедряя серьезный менеджмент проектов. Можно встретить убеждение – “Не думай как лучше, просто сделай этот сайт.” Это целесообразно, если это позволит выдавать более качественный продукт за меньшее время. Что, соответственно, приведет к снижению расходов. Плюс к тому - бонус в качестве более спокойной, предсказуемой работы – а это, как мне кажется, один из ключей к снижению текучести кадров.

А может быть, дело и в том, что заказчики еще не готовы платить по нескольку десятков (сотен) тысяч долларов за качественную разработку. И правда, зачем, если знакомые студенты сделают ту же работу, только за тысячу - полторы…

6.2. Типичные менеджерские ошибки, совершаемые заказчиком при разработке сайта

- **Отсутствие четкого осознания целей проекта**

Очень часто при создании сайта у заказчика отсутствует четкое понимание целей, которые стоят перед проектом. И в итоге результат может оказаться плачевным: вы “что-то” попросили, мы “что-то” и

сделали. Четкое понимание, что сайт нужен только для галочки и указания на визитках (а для многих компаний это так и есть) избавило бы множество заказчиков от головной боли и лишних трат.

Помните, что сайт – это не набор красивых изображений в Интернете, а, прежде всего, маркетинговый инструмент, который должен выполнять свое предназначение, а не “просто быть”.

Крайне рекомендуется перед началом проекта провести предпроектное исследование и заказать разработку веб-стратегии (о ней подробнее в пункте про типовой цикл проекта), которая позволит сформировать некое видение проекта на всех этапах его жизненного пути.

- **Неправильная оценка и распределение бюджета проекта**

Довольно часто заказчик не может правильно оценить бюджет проекта и правильно расставить приоритеты по аспектам работ. Очень часто заказчик исходит из личных предпочтений и устоявшихся стереотипов (например, “дизайн сайта не важен, главное - информация”). Это неверно, бюджет должен формироваться исходя из четкого понимания целей и задач проекта. Если вы определили, что основная задача сайта – поддержание бренда или определенной марки продукта в Сети, глупо экономить на разработке дизайн-концепции.

Типичная ошибка – большой бюджет на продвижение проекта (допустим, сотни тысяч долларов в год) - и мизерный на сам сайт. Подумайте, вкладывая такие деньги в рекламу, логично потратить немного больше денег на разработку, чтобы посетители сайта не были разочарованы. Верно и обратное: не имеет смысла раздувать бюджет на сайт, если никаких целей, кроме, собственно, его наличия нет.

Если у вас есть сомнения в собственных возможностях определения общего бюджета или распределения его по составным частям, обратитесь за консультацией во внешнюю экспертную организацию (только не к подрядчику). Желательно, чтобы эта организация не занималась разработкой сайтов, а специализировалась именно на консалтинге, иначе ее анализ может быть предвзятым.

- **Затягивание согласования работ**

Главная причина затягивания сроков со стороны заказчика – мучительные согласования результатов работ. Особенно это касается разработки дизайн-концепции сайта. Чем больше людей со стороны заказчика участвуют в этом процессе (а по дизайну каждый считает себя специалистом) – тем больше хаоса привносится в проект. Это

необходимо исключить – у проекта должен быть менеджер, курирующий все вопросы по проекту, а так же четко выделенный человек, принимающий окончательные решения. Разработчику обязательно нужно обеспечить доступ как к менеджеру, так и принимающему решения лицу, как бы трудно это ни было. Остальных “советчиков” необходимо исключить из процесса, потому что выполнения всех их пожеланий и комментариев (зачастую необоснованных и противоречащих друг другу) затянет проект на неопределенный срок. Это ни в коей мере не касается этапа тестирования сайта и выявления ошибок – тут нужно привлечь максимально широкий круг людей, чтобы вычистить все огрехи. Нужно четко построить модель взаимодействия по проекту с разработчиком именно на уровне менеджмента компании.

При принятии решения по дизайну необходимо **максимально абстрагироваться от собственных предпочтений** и мыслить в категориях пользователя и общей адекватности предлагаемых вариантов.

Очень часто заказчик начинает “придираться” к мелочам в дизайне, хотя на самом деле ему не нравится предложенный концепт в принципе, но он просто не может (или опасается) это выразить. Важно **отследить этот момент и предложить разработать новый вариант**. Разработчику это будет проще, чем потратить месяцы на согласование не играющих роли мелочей, которые все равно не добавят удовлетворенности заказчику. Не теряйте за мелочами общего видения проекта!

Еще один важный аспект – не пускайте проект на самотек. Очень часто после первого срыва сроков по этапам, заказчик расслабляется и перестает оперативно реагировать на ситуацию. Это может привести к огромным задержкам по срокам, поскольку разработчик часто склонен сделать то же самое. Проект может попасть в “мертвую точку”, когда все потеряли к нему интерес. Сдвинуть ситуацию с этой точки может стоить огромных усилий.

- **Неправильная организация этапа подготовки материалов на сайт**

Второй по важности аспект, влияющий на затягивание сроков – неправильная организация процесса подготовки материалов на сайт. Очень часто заказчик подходит к этому процессу слишком оптимистично, обещая предоставить материалы в кратчайший срок. Это иллюзия. Как правило, подготовка таких материалов –

дополнительная обязанность людей, занимающихся чем-то внутри компании.

Постарайтесь выделить какое-то время сотрудников непосредственно на подготовку материалов на сайт. Установите жесткий дедлайн и примите важное решение – если контент не будет готов к обозначенному сроку, то он не будет готов никогда. Со стороны разработчика будет также абсолютно оправдано установить некий срок, после которого он не будет принимать от вас никаких материалов, и их отсутствие не будет влиять на процесс согласования его работы.

Соберите готовый материал и закажите копирайтинг недостающего, если это необходимо (крупные студии-разработчики с радостью предложат вам свою помощь). Не забывайте предоставлять разработчику максимально полную информацию, он всегда сможет отмести лишнее.

“Хорошая ситуация” – когда на момент старта проекта в студию приезжает менеджер со стороны клиента и привозит кипу бумаг и высокую стойку CD с информацией (все-все, что он смог собрать). Это правильно.

Разработчик не обманывает вас, когда говорит, что материалы крайне важны для его работы. Это действительно так. Идеальная ситуация – 100%-я готовность материалов на момент начала работ по дизайну. Наличие материалов к этому моменты позволит вам существенно ускорить процесс, а также лишить разработчика типичной отговорки “не было материалов, поэтому задержали сроки”, хотя, как правило, это не является отговоркой, а действительной причиной.

- **Отсутствие дедлайна работ**

Жесткий дедлайн, о котором известно вам и разработчику – отличное средство мотивации деятельности по проекту с обеих сторон. Его наличие позволяет отмести мелочи и долгое согласование незначительных моментов, сформировать нацеленность на результат. Если жесткого дедлайна нет – придумайте его (вплоть до указания финансовых санкций в случае его несоблюдения). Отсутствие дедлайна расслабит как вас, так и разработчика.

- **Оптимизм исполнителя при составлении календарного плана**

Довольно часто при утверждении календарного плана разработчик рассматривает “идеальную ситуацию” и указывает

слишком малые сроки. Идеальных проектов не бывает, это надо помнить. Календарный план нужно оценить на общую адекватность, обязательно проверить на наличие этапов на согласование работ и подписание документов, получение оплаты и пр.

Кроме того, нужно помнить, что малые сроки в плане – конкурентное преимущество разработчика, и он мог намеренно занизить их на стадии коммерческого предложения. С этим можно бороться прописыванием жестких санкций в договоре, а также предложением пересмотреть первичный план, например, по итогам составления ТЗ.

- **Нарушение типового цикла проведения работ**

Очень часто при работе над сайтом нарушается типовой цикл разработки проекта. Это может происходить как по инициативе разработчика, так и заказчика. Процесс по возможности должен идти последовательно по циклу. Поскольку очень часто результат по предыдущему этапу влияет на следующий, необходимо продолжать работы, только согласовав и утвердив предыдущий этап.

Например, заказчик часто требует начать параллельно с этапом дизайна этап сборки и верстки сайта. Это в корне неверно, поскольку при внесении изменений в дизайн-макеты может поменяться вся концепция ресурса, и все выполненные работы по дальнейшим этапам пойдут насмарку.

Типовой цикл прохождения проекта выглядит примерно так:

- Определение целей и задач проекта
- Определение позиционирования, анализ информации о продукции/услугах и целевой аудитории
- Разработка общей веб-стратегии компании (проекта).
- В веб-стратегию должна войти информация о целях и задачах ресурса, целевой аудитории, проведен анализ конкурентов, должны присутствовать рекомендации по структуре и функционалу сайта, а также разработан примерный план дальнейшего продвижения ресурса (анализ действий конкурентов, примерные мероприятия и бюджеты).
- Составление подобной стратегии существенно помогает на всех стадиях жизненного цикла вашего проекта. Имеет смысл заказывать разработку стратегии внешней по отношению к вашему подрядчику экспертной структуре.
- Разработка Технического Задания (ТЗ) на сайт, итоговой сметы и календарного плана работ (задача ТЗ – максимально

подробно определить все аспекты работ по сайту, создать единое видение (это очень важно) проекта и заказчика и исполнителя).

- Разработка дизайн-концепции сайта
- Разработка макета главной страницы
- Разработка макетов внутренних страниц
- HTML-верстка сайта
- Разработка анимационных FLASH-элементов
- Сборка сайта на базе CMS-системы и разработка дополнительного функционала
- Контент-наполнение сайта, пакетный внос данных в БД
- Запуск пилотной версии сайта, тестирование, устранение ошибок
- Перенос сайта на хостинг, тестирование, открытие сайта

Обратите внимание, в данном цикле не указаны моменты, связанные с подготовкой необходимой информации и согласований этапов работ.

Необходимо помнить, что процесс разработки сайтов подчиняется общепринятым принципам ведения проекта и разработки ПО. Согласно этим принципам, рекомендуется по возможности вести процесс итерационно, корректируя собственные действия по мере приобретения опыта и новых знаний по проекту. Работая над большим проектом, крайне полезно создать и отладить работу облегченной версии, постепенно дорабатывая новый функционал.

- **Создание “сайта-памятника”**

Еще одна типовая ошибка заказчика – считать, что после создания сайта его работа по развитию Интернет-направления заканчивается. Это в корне не верно. Создание сайта – точка “ноль” для его существования. Чтобы сайт стал эффективным маркетинговым инструментом, необходимо разработать стратегию его продвижения, а также на постоянной основе заниматься его обновлениями, мониторингом состояния, произведением улучшений и доработок, развитием.

В сети существует множество примеров красивых “сайтов-памятников”, которым не уделяется внимания со стороны их владельцев. Они висят “мертвым грузом”, не принося никакой пользы и сводя к нулю все усилия, потраченные на их разработку.

Небольшой пример – большинство заказчиков хотят видеть на своем сайте ленту новостей, которая публикуется на главной странице. При этом у большинства компаний не хватает сил

поддерживать ее в актуальном состоянии (или просто нет должного количества информационных поводов). Определите этот момент еще на стадии проектирования, ведь нет ничего хуже сайта с новостями за прошлый год на первой странице – он моментально создает у пользователя ощущение “заброшенности ресурса”. Лучше не публикуйте новости вообще, они, как правило, не очень интересны пользователю.

- **Регистрация домена студией или лично менеджером проекта**

Зачастую такой вопрос, как регистрация домена, отдается на откуп разработчику, и разработчик регистрирует домен на себя. В случае возникновения конфликтной ситуации (особенно на стадии, когда проект уже запущен и имеет какой-то индекс цитируемости, а почта и адрес сайта указаны в рекламной продукции). Компания подрядчик может разыграть этот козырь, угрожая снять сайт и почтовый сервер с вашего домена. Это же относится и к регистрации домена лично на имя менеджера с вашей стороны. Если отношения не сложатся, он сможет забрать его себе, оставив вас ни с чем. Поэтому надо обязательно проконтролировать, чтобы домен был зарегистрирован именно на вашу компанию как на юрлицо.

- **Большая предоплата фрилансеру**

Работая с фрилансером, необходимо помнить. Что получение большой предоплаты крайне расслабляет человека и сводит его мотивацию практически к нулю. Этот грустный факт подтвержден многолетним опытом. Не давайте предоплату частнику более 10-20% процентов, даже если вы очень в нем уверены. Это не касается работы с крупными студиями – получение предоплаты в 40-60% процентов от бюджета проекта является нормой для рынка.

- **Отсутствие контроля откатов**

Не надо забывать о существующей повсеместно практике откатов. Если ваш менеджер, которому вы поручили создание сайта, настойчиво рекомендует конкретного исполнителя, и у вас появляются сомнения в разумности представленной сметы, ее необходимо проверить. Отправьте запрос в 2-3 компании, работающие в диапазоне вашего общего бюджета, и сравните их с предлагаемой сметой. Возможно, вы будете неприятно удивлены.

- **Статика или динамика: что выбрать с учетом развития сайта**

Все сайты в сети можно разделить на две большие группы: статические и динамические сайты. С точки зрения посетителя сайта порой не важно, на какой странице он находится, статической или динамической, иногда даже трудно точно это определить. Но я хочу рассмотреть такое разделение с точки зрения разработчика - создателя сайта.

Статическими являются страницы, которые целиком хранятся на сервере и показываются посетителю в своем неизменном виде (следует учесть, что статическая страница может содержать некоторые изменяемые элементы, например баннеры, однако она все равно остается статической).

Статическим называется сайт, большинство или все страницы которого являются статическими. Таких сайтов довольно много, с них, можно сказать, начинался интернет.

Динамическими являются страницы, формируемые сервером из нескольких частей или получаемые путем внесения либо изменения данных в страницу.

Она собирается несколькими различными способами из данных, хранящихся на сервере, и только после этого показывается посетителю.

Первым вариантом может быть объединение нескольких (двух и более) отдельных частей в одну страницу - это самый простой способ генерации.

Вторым вариантом является заполнение шаблонной страницы какой-либо информацией, хранящейся отдельно или получаемой в результате работы алгоритма (например, в результате вычислений).

Третьим, и, пожалуй, самым распространенным вариантом является сочетание первых двух во всевозможных вариациях, т.е. страница собирается из нескольких кусочков, в которые при этом вносятся различные изменения.

Динамический сайт похож на компьютерную игру. В ней есть определенный сценарий, свои персонажи и интерьеры, но финальная картинка получается только после совмещения всех этих частей, причем не без участия пользователя.

6.3. Преимущества и недостатки динамических сайтов

Напрашивается вопрос, а зачем вообще это нужно, делать страницу динамической, разбивать ее на части, хранить информацию где-то отдельно и т.д.? Не проще ли создать сайт таким, как его должен видеть посетитель?

Как говорилось ранее, со статических сайтов начинался интернет, динамические страницы и сайты появились позднее, но начали теснить своих прародителей, а это значит, что они имеют свои преимущества. Вот давайте и рассмотрим, какие преимущества имеют динамические сайты по сравнению со статическими, ну а чтобы обзор был полным, уделим время и недостаткам.

Хочу сразу заметить, что, давая определения, я начинал с описания отдельных страниц. Это было сделано для того, чтобы вам было понятнее, о чем идет речь. Теперь же, при сравнении, я буду рассматривать целые сайты: статические и динамические. Соответственно и преимущества будут касаться именно готовых сайтов, т.к. подчас для страниц они просто не подходят.

Итак, с появлением языков программирования, выполняемых на стороне сервера, появилась возможность вносить изменения в данные отправляемые посетителю. Что же это дало?

- Разделение информации и дизайна
- Упрощение модификации и обновления страниц
- Возможность изменять контент, реагируя на действия посетителя

Давайте рассмотрим каждый пункт поподробнее.

- **Разделение информации и дизайна сайта**

Использование динамических страниц позволяет хранить некий шаблон дизайна, в который, в зависимости от страницы, на которую зашел посетитель, помещается необходимое наполнение. Такой вариант очень удобен, ведь для всего сайта создается один или несколько шаблонов, и все изменения дизайна, которые требуется сделать на сайте, производятся только с ними.

В качестве примера представьте себе корпоративный сайт, на котором несколько сотен страниц (это не так уж много по нынешним меркам). И предположим, организация решила поменять свой логотип, нет ничего проще - если сайт статический, нужно внести изменения в каждую страницу. А теперь представьте, что сайт

состоит из тысяч страниц, а изменения приходится делать постоянно. Сколько временных и человеческих ресурсов для этого потребуется?

Если же сайт динамический, все гораздо проще. Изменения вносятся в один или несколько шаблонных файлов, и все страницы сайта автоматически изменяются.

Данный подход также позволяет разграничить полномочия людей, занимающихся наполнением и модификацией сайта. Т.е. одни сотрудники могут заниматься дизайном, другие наполнением страниц. В идеале они даже не будут пересекаться, только посетитель сайта будет видеть результат их совместной работы на экране своего монитора.

С точки зрения организации работы и разделения труда, вариант создания сайта на основании шаблонов практически идеален. Человек, ответственный за дизайн сайта не сможет вмешаться в процесс наполнения страниц контентом, и за все недочеты в дизайне будет нести ответственность только он. И наоборот, тот, кто занимается информационным наполнением сайта, не сможет нарушить дизайн сайта. Каждый занимается своим делом, не мешая другим. Это ускоряет работу и снижает затраты.

- **Упрощение модификации и обновления страниц на сайте**

Разделение информации и дизайна на сайте позволяет ускорить процесс обновления и наполнения сайта, т.к. не требует от людей, выполняющих эти операции, знаний в областях html-разметки, графики и т.д. Т.е. можно даже подготовить дизайн сайта (заказать у фирмы, предоставляющей такие услуги), а потом производить наполнение сайта собственными силами, не затрачивая средства на постоянное привлечение сторонних или включение в свой штат специалистов по дизайну, что потребовалось бы при статической организации сайта. Да и временные затраты на обновление значительно сокращаются.

- **Возможность изменять контент сайта, реагируя на действия посетителя**

Третье достижение стало прорывом на пути развития глобальной сети. Посудите сами, все преимущества, описанные в первых двух пунктах, облегчали работу создателей сайтов и снижали их расходы, но для посетителей это было не очень важно. А вот третье преимущество отразилось непосредственно на них. Только благодаря ему вы можете видеть огромное количество электронных магазинов, виртуальных клубов, интернет-игр и т.п. Только

возможность изменять наполнение сайта под конкретного посетителя позволило воплотиться в жизнь этим проектам.

Еще один пример - это форумы и чаты, коих на просторах сети развелось бесчисленное множество. Все они создаются динамически, без поддержки этой технологии ни один из них не смог бы существовать. Доски объявлений, клубы по интересам, виртуальные игры и соревнования - ничего этого бы не смогли увидеть.

Недостатки динамических сайтов

Как и у всего остального в нашем неидеальном мире, у динамических сайтов есть свои недостатки.

- **Первым недостатком** является необходимость использования дополнительных программных средств для построения динамического сайта. На статическом сайте все страницы уже готовы, серверу остается только показать их посетителю, а на динамическом сайте необходимо вносить в них какие-то изменения, для этого требуется соответствующие программные решения.

В зависимости от сложности сайта, трудоемкость и стоимость разработки таких программ может очень сильно варьироваться. Сейчас существует множество готовых решений для создания сайта, в том числе и бесплатных. В сети можно найти не один десяток всевозможных бесплатных скриптов, которые позволят вам создать на сайте форум, доску объявлений, клуб знакомств, магазин и т.д. Но если требуется что-то специфическое, здесь не обойтись без дополнительных разработок.

- **Вторым недостатком** является повышение требований к аппаратным мощностям серверных систем. Этот недостаток непосредственно следует из предыдущего, т.к. теперь серверу требуется еще выполнить какую-то программу для модификации страницы сайта, а только потом выдать ее посетителю. Особенно заметной эта проблема становится на сайтах с большой посещаемостью. Часто в таких случаях приходится производить дополнительные оптимизации кода для нормальной работы сайта.

Следовательно, стоимость услуг, по содержанию такого сайта, намного выше, нежели статического. Хотя сейчас, даже многие бесплатные хостинги поддерживают возможность создания динамических сайтов, не говоря уже о платных, где все необходимое входит в стандартный набор услуг.

- **Третьим недостатком**, также вытекающим из первого, является сложность больших структурных изменений сайта.

Представьте, у вас на сайте размещен форум, а вам захотелось, чтобы был еще и чат. Если этой возможности изначально не было, то, как бы вы не меняли дизайн или наполнение форума, он не сможет превратиться в чат. Вам для этого придется изменить программу.

Несмотря на эти недостатки, динамических сайтов в сети становится все больше и больше, видимо, описанные мной ранее преимущества перекрывают все недостатки. Теперь давайте рассмотрим, что же на сегодняшний момент имеется в сети, какие возможности может получить человек, желающий создать сайт.

Системы управления контентом

Технологии не стоят на месте, развиваются и возможности, предлагаемые для создания сайтов. Сейчас все большую популярность завоевывают системы управления контентом. Что это такое? Давайте разберемся.

CMS - это аббревиатура от **Content Management System**, что в дословном переводе - Система Управления контентом сайта. Проще говоря, это тот самый программный комплекс, который позволяет вам изменять дизайн и наполнение сайта таким образом, как вам требуется.

Сейчас в сети можно найти множество таких систем, какие-то из них бесплатные, какие-то платные. Часто фирмы-разработчики предоставляют своим клиентам такие системы. Каждая система индивидуальна и обладает своими достоинствами и недостатками.

Преимуществом визуальной системы редактирования является то, что вы сразу видите, как будет выглядеть та или иная страница на сайте. Вам не нужно задумываться о том, как данные хранятся в системе, как собирается страница, достаточно наполнить ее информацией и сайт готов.

Такое решение очень выгодно для небольших компаний, которые не могут позволить себе держать в штате программиста, дизайнера, верстальщика, достаточно только оператора для ввода и обновления информации. Такой вариант выгоден и частным лицам, желающим создать сайт, но не имеющим в этом большого опыта.

Также данная технология предусматривает возможность создания интернет-магазина, на базе конструктора для создания сайта

Не стоит считать, что динамические сайты - это всегда правильно решение. Всегда следует исходить из ситуации. Если нет необходимости динамического изменения данных, а сайт предполагается не очень большим, можно обойтись и статическими

страницами. Ведь бывают одностраничные сайты, так зачем под них писать программы, когда проще создать статическую страницу. Но не во всех случаях это возможно.

Следует исходить из целесообразности использования того или иного средства для создания сайта. Здесь следует учитывать как первоначальные, так и последующие финансовые и трудовые затраты, необходимые на поддержание сайта.

Вопросы для самоконтроля:

1. Назовите распространенные проблемы, возникающие при при управлении веб-проектами.
2. Назовите типичные менеджерские ошибки, совершаемые заказчиком при разработке сайта
3. Назовите достоинства и недостатки динамических сайтов.
4. Что такое система управления контентом?

7. ЭТАПЫ РЕАЛИЗАЦИИ ВЕБ-ПРОЕКТА

Процесс веб-разработки включает в себя следующие этапы:

- Постановка задачи / разработка концепции проекта
- Проектирование
- Маркетинг
- Создание макетов дизайна
- Верстка макетов
- Программирование
- Тестирование, исправление багов
- Поддержка
- Постановка задачи / разработка концепции проекта

Первый этап – наиболее ответственный, т.к. от его успешного выполнения зависит дальнейшая судьба веб-проекта. На этом этапе собирается вся необходимая информация, которая только может потребоваться для его успешной реализации. Этот этап строится на непосредственной коммуникации заказчика и исполнителя.

У заказчика уточняются цели, пожелания и мысли относительно его проекта, поэтому заказчик должен как можно четче, яснее и информативнее ответить на все задаваемые вопросы – это в его интересах.

Обычно уточняются следующие аспекты:

Задача. Какая задача поставлена перед сайтом? Вы хотите продавать какую-либо продукцию, услуги или предоставлять информацию по какой-либо теме?

Цели. Чего Вы хотите добиться, благодаря новому сайту? 2 наиболее популярные цели: распространение информации и получение прибыли.

Целевая аудитория. Кто относится к Вашей целевой аудитории? Постарайтесь как можно точнее обрисовать портрет представителя Вашей целевой аудитории.

Контент. За какой информацией будут обращаться представители Вашей целевой аудитории к Вашему сайту? Может быть, им будет нужна практическая информация или информация об интересующих их товарах или услугах…?

Детальный анализ данных, построение логической диаграммы входящих-выходящих данных;

Утверждение платформы, используемых языков программирования;

Предварительное планирование трудовых ресурсов (обсуждается наличие и квалификация специалистов), также определяется наличие специалистов по поддержке проекта после сдачи его заказчику.

Общие рекомендации по срокам исполнения проекта

Определение стороны, ответственной за публикацию и продвижение проекта в сети. В случае, если это входит в задачи разработчиков, разрабатывается заранее приблизительный план маркетинговой кампании.

Проектирование

На этом этапе используем информацию, полученную на предыдущем этапе, для проектирования будущего сайта. Сейчас мы должны определить все необходимые разделы и подразделы сайта, а также их содержимое. Необходимо помнить о том, что сайт должен быть юзабельным, а его информационная архитектура понятной и прозрачной.

Особое внимание при проектировании следует уделить таким интерактивным элементам как формы, корзина, flash и т.д.

На этом этапе обычно происходит следующее:

- структура данных, определение связей между таблицами базы данных, структуры передачи данных из внешних источников;
- определение уровня автоматизации обработки данных, разработка структуры управления данными;
- проводится спецификация форм и порядок их появления;
- структура пользовательского интерфейса: пункты меню и элементы навигации, необходимые уровни вложенности;
- разработка эскизов дизайна проекта (количество вариантов утверждается заранее).

Маркетинг

Маркетинг – в нашем случае, это исследование рынка, компании заказчика и конкурентной среды. Не все заказчики до конца понимают, зачем нужен данный этап и часто полагают, что это пустая трата времени и денег. Конечно, не для всех проектов маркетинг обязателен. В каких случаях оно действительно необходимо, а в каких – факультативно? Чтобы ответить на этот вопрос, нужно для начала оценить масштабы самого проекта и его прямые цели.

Маркетинг необходим:

- При продвижении/продаже продукта
- При привлечении новых потребителей
- Для решения задач брендинга
- Для любой другой коммерчески значимой деятельности в Интернете

Характерные задачи на этапе:

- Анализ источников информации
- Описание целевой аудитории
- Анализ спроса
- Определение структуры рынка
- Семантическое ядро
- Определение схемы сайта
- Составление рекомендаций
- Создание макетов дизайна

Дизайн по праву считается одним из наиболее важных и ответственных этапов работы по созданию сайта, т. к. не только привлекает внимание потенциальных заказчиков компании, но и реализует имидж фирмы в среде Интернет.

На этом этапе разрабатываются концепции дизайна, графические элементы, которые призваны сделать сайт непохожим на другие.

К дизайну предъявляются следующие требования:

- Оригинальность
- Технологичность
- Функциональность

Дизайн сайта должен подчиняться одной цели, единой концепции. Он должен адекватно отражать характер деятельности заказчика, органично сочетаться с информационным наполнением, структурой сайта и принципами организации информации.

Наиболее важным фактором, на который необходимо опираться при создании дизайна – портрет представителя целевой аудитории сайта. Например, одинаково ли должны выглядеть сайты, целевая аудитория которых в первом случае молодежь, во втором – бизнесмены? Конечно, нет.

Дизайнер должен создать несколько различных макетов дизайна и предоставить их на суд заказчику. Готовый макет должен быть в виде простой .jpg картинки.

После согласования одного из макетов, начинается его доведение до ума (если требуется).

Верстка макетов

Верстка представляет собой процесс интеграции текстового содержания, графики и программных компонентов в единое целое, т. е. придание страницам окончательного вида. В процессе верстки страницы приобретают вид, в котором они предстанут перед конечным пользователем (за исключением информационного наполнения).

Именно на этом этапе начинается непосредственная материализация будущего веб-сайта. К этому моменту макет дизайна сайта уже утвержден. На этом этапе верстаются (по веб-стандартам XHTML+CSS) макеты главной и необходимых внутренних страниц, а также пишется система управления контентом, отвечающая заданным в техническом задании требованиям.

В это время происходит дополнительный контроль качества выполняемой работы, производится оптимизация web-страниц под особенности конкретных браузеров, используемых посетителями сайтов для навигации по Интернету. Учитываются особенности представления страниц при различных настройках глубины цвета и экранных разрешений.

Как только работа над этим этапом завершается, начинается работа над следующим этапом…

Программирование

На данном этапе происходит разработка и подключение программных компонентов сайта, призванных обеспечить посетителей необходимыми функциональными возможностями. Этот этап является наиболее сложным по реализации в технологическом плане. Большинство решений, разрабатываемых на данном этапе, основываются на технологиях работы с базами данных и на построении динамически генерируемых страниц сайта на основе информации, содержащейся в базе данных сайта.

Синхронизация

Синхронизация – это соединение сверстанных и черновых шаблонов сайта. Этот этап начинается только после завершения верстки и программирования. Если макеты и техническое задание были согласованы и утверждены заказчиком, то проблем на этом этапе не возникнет. Верстальщик должен получить от программиста

хорошо документированные гибкие черновые шаблоны. Рекомендуется пользоваться технологиями xml/xslt.

На данном этапе будет задействован следующий персонал: верстальщик, программист, менеджер проекта.

Тестирование, исправление багов

На этом этапе проводится детальное тестирование сайта, тестирование на идентичное отображение сайта во всех распространенных веб-браузерах, работоспособность всех форм, скриптов, flash-роликов и т.д.

В случае нахождения каких-либо ошибок, недочетов и прочих багов, они в немедленном порядке исправляются.

Как только сайт был тщательно и неоднократно проверен, его можно загружать на клиентский FTP, а также переходить к заключительному этапу…

Публикация

Публикация — это обеспечение хостинга интернет-сайта и «привязка» сайта к предварительно зарегистрированному доменному имени. Обычно, содержимое сайта загружается на интернет-сервер через протокол FTP. В результате, интернет-сайт становится доступным всем пользователям сети Интернет.

Доменное имя — это адрес сайта в сети Интернет. Учитывая специфику деятельности и название компании, обычно составляется список наиболее подходящих доменных имен, и также проверяется их «занятость».

Хостинг — это размещение интернет-сайта на подключенном к сети Интернет сервере с соответствующим набором программного обеспечения, необходимого для корректной работы интернет-сайта. Заказчик сайта зачастую имеет на примете хостинг и после создания сайта просит, чтобы сайт разместили туда. Хотя бывает, что хостинг подыскивает фирма-разработчик веб-сайта.

Наполнение контентом

Контент – это содержимое сайта, та информация, которая может быть полезной его посетителям. Наполнение контента — это внесение графической и текстовой информации в шаблонные web-страницы сайта в рамках, предусмотренных его дизайном.

Наполнение сайта возможно через административный интерфейс (систему управления сайтом при ее наличии) либо путем заполнения статичных страниц после верстки сайта.

Перед тем, как наполнять сайт статьями, необходимо определить перечень ключевых слов по выбранной тематике. Необходимо разделить выбранные ключевые слова на тематические группы, создать на сайте разделы под каждую из таких групп. Статьи, которые будут публиковаться, должны содержать по возможности как можно большее количество определенных выше ключевых слов.

Очень желательно, чтобы контент на сайте был уникальным, то есть таким, которого нет ни на одном другом сайте. В этом случае этот сайт будет гораздо выше ранжироваться поисковыми системами.

Способов добычи уникального контента довольно много. К самым распространенным относятся:

- копирайтинг (самостоятельное написание статей),
- рерайтинг (изменение чужих статей до неузнаваемости с сохранением смысла),
- переводы иностранных статей.
- Если этим заниматься нет возможности – можно воспользоваться услугами субподрядчиков или специальных сервисов.

Поддержка

Поддержка сайта – широкое понятие, включающее в себя как поддержание работоспособности сайта, так и его последующее развитие.

Еще на этапе проектирования необходимо задаться вопросом, каким образом будут добавляться новые разделы и материалы, что будет происходить со старыми. Возможно, потребуется создание архива новостей, куда будут попадать новости, потерявшие свою актуальность.

Еще более важной является постоянная актуализация информации на сайте (частота обновления, ответственный за наполнение сотрудник). Важно не забывать посетителях сайта, поэтому им должна предоставляться больше новой информации.

Продвижение (раскрутка)

Готовый, наполненный информацией сайт еще не гарантирует вам приток новых клиентов. Главная причина этого — пользователи Интернет могут не встретить ваш сайт в сети. По статистике большинство пользователей находят информацию и приходят на сайты через поисковые системы. Именно поэтому популярность ресурса зачастую зависит от его позиции по результатам поиска.

Чтобы ваш сайт занимал лидирующие позиции в поисковых системах необходимо провести ряд мероприятий по его продвижению в сети.

Продвижение — это своего рода «рекламная кампания» по узнаванию сайта и повышения его посещаемости. Сюда входит:

регистрация сайта в поисковых системах

обмен ссылками

баннерная реклама

и т.д.

Вопросы для самоконтроля:

1. Назовите основные этапы реализации веб-проектов.
2. Что такое контент сайта. Какие способы получения уникального контента существуют.
3. Что такое продвижение сайта.

8. РАБОТА В СРЕДЕ MICROSOFT PROJECT

8.1. Обзор интерфейса инструмента управления проектами Microsoft Project

Предлагаемые Microsoft инструменты управления проектами предусматривают наличие следующих вариантов конфигурации MS Project 2007:

• **Microsoft Office Project Standard 2007** — новая версия настольного приложения для индивидуального планирования и управления проектами. Она содержит набор базовых функций и предназначена в первую очередь для отдельных пользователей или небольших коллективов, не использующих для обмена данными по проекту сетевые технологии.

• **Microsoft Office Project Professional 2007** — существенно обновленное настольное приложение, ориентированное на применение в организациях, в которых требуются возможности управления проектом (или портфелем проектов) на уровне предприятия. Project Professional содержит помимо функций, реализованных в Project Standard, рад дополнительных возможностей. Они относятся в первую очередь к организации совместной работы над проектом на основе сетевых технологий.

• **Microsoft Office Project Server 2007** — это продукт, который служит платформой для организации совместной работы над проектом на уровне предприятия. MS Project Server обеспечивает централизованные настройки для пользователей, единый пул ресурсов, Web-интерфейс для совместной работы участников проекта, а также содержит средства OLAP-анализа и моделирования портфеля проектов. Хранение данных осуществляется в СУБД Microsoft SQL Server. Для OLAP-анализа используется служба Microsoft SQL Server Analyses Services.

• **Microsoft Office Project Web Access 2007** — составная часть сервера MS Project Server, обеспечивающая реализацию Web-интерфейса. Обеспечивает участникам проектов доступ к проектной информации через веб-браузер Internet Explorer, совместное управление документами, вопросами и рисками проектов.

Композиция из трех продуктов — **Microsoft Office Project Professional 2007, Microsoft Office Project Sewer 2007 и Microsoft Office Project Web Access 2007** — представляет собой не что иное,

как корпоративное решение для управления проектами — Microsoft Office Enterprise Project Management 2007.

Необходимо отметить, что все программные инструменты управления проектами (в том числе и MS Project) изначально **не предназначены для автоматической генерации оптимальных управляющих решений**. Их следует рассматривать и использовать как средства поддержки принятия решений менеджером проекта: с помощью MS Project менеджер может буквально за считанные минуты оценить эффективность нескольких альтернативных вариантов реализации проекта и выбрать стратегию, в наибольшей степени отвечающую интересам компании и целям проекта.

Именно поэтому при разработке MS Project 2007 был реализован подход, позволяющий усилить аналитические способности пакета за счет более тесной интеграции со специализированными средствами аналитической обработки данных.

Например, для анализа проектных данных вы можете использовать такие инструменты, как мощное средство графического представления статистики — MS Data Analyzer, MS Business Intelligence Portal, который позволяет получить через Web бесплатный доступ к функциям статистической обработки.

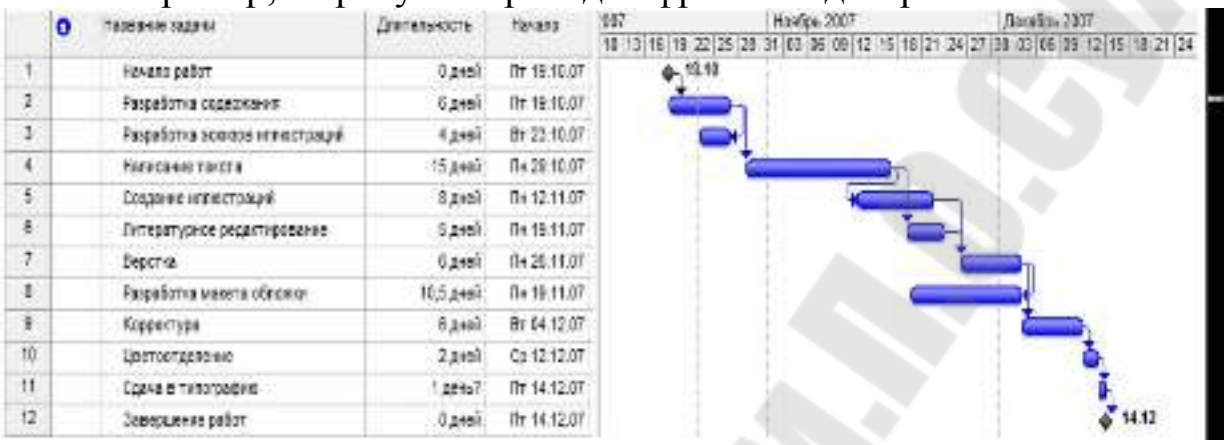
Применение MS Project на стадии планирования поможет руководителю ответить на следующие вопросы:

- Насколько вообще реально воплощение в жизнь данного проекта?
- Какие конкретно работы необходимо выполнить для достижения целей проекта?
- Какой состав исполнителей, соисполнителей и какие виды материальных ресурсов потребуются для реализации проекта?
- Какова стоимость проекта и как наиболее выгодно распределить во времени финансовые затраты на реализацию проекта? Кто должен отвечать за те или иные виды работ?
- Насколько велик риск и каков возможный ущерб при завершении проекта на той или иной стадии?

Для ответа на первый вопрос требуется провести полный анализ проекта по методу критического пути с использованием ресурсного планирования, однако без излишней детализации. В этом отношении весьма большую помощь могут оказать шаблоны, входящие в состав стандартной конфигурации MS Project. Каждый из шаблонов относится к определенной сфере, и может считаться своеобразным

стандартом соответствующего плана проекта. Внеся в него необходимые коррективы в соответствии с особенностями конкретного проекта, можно получить вполне реалистичную оценку возможного развития событий и требуемых затрат.

Например, на рисунке приведен фрагмент диаграммы Ганта.



	Название задачи	Длительность	Начало
1	Начало работ	0 дней	Пт 19.10.07
2	Разработка содержания	6 дней	Пт 19.10.07
3	Разработка эскизов иллюстраций	4 дней	Вт 23.10.07
4	Написание текста	15 дней	Пн 29.10.07
5	Создание иллюстраций	8 дней	Пн 12.11.07
6	Литературное редактирование	5 дней	Пн 19.11.07
7	Верстка	6 дней	Пн 26.11.07
8	Разработка макета обложки	10,5 дней	Пн 19.11.07
9	Корректура	6 дней	Вт 04.12.07
10	Цветоотделению	2 дней	Ср 12.12.07
11	Сдача в типографию	1 день?	Пт 14.12.07
12	Завершение работ	0 дней	Пт 14.12.07

Рис. 8.1. Диаграмма Ганта

Ответ на второй вопрос также может быть получен с помощью одного из стандартных расписаний. Если же подходящего шаблона для планируемого проекта нет, то структуру проекта придется создавать вручную. Тем не менее и в этом случае MS Project 2007 способен оказать существенную помощь, поскольку в его составе имеются средства построения сетевого графика (Network Diagram). Для работ проекта автоматически устанавливаются параметры, заданные по умолчанию (такие как длительность, календарные даты начала и окончания и т. д.). На рисунке показан один из возможных вариантов представления сетевого графика, наиболее близкий его «бумажному» аналогу.

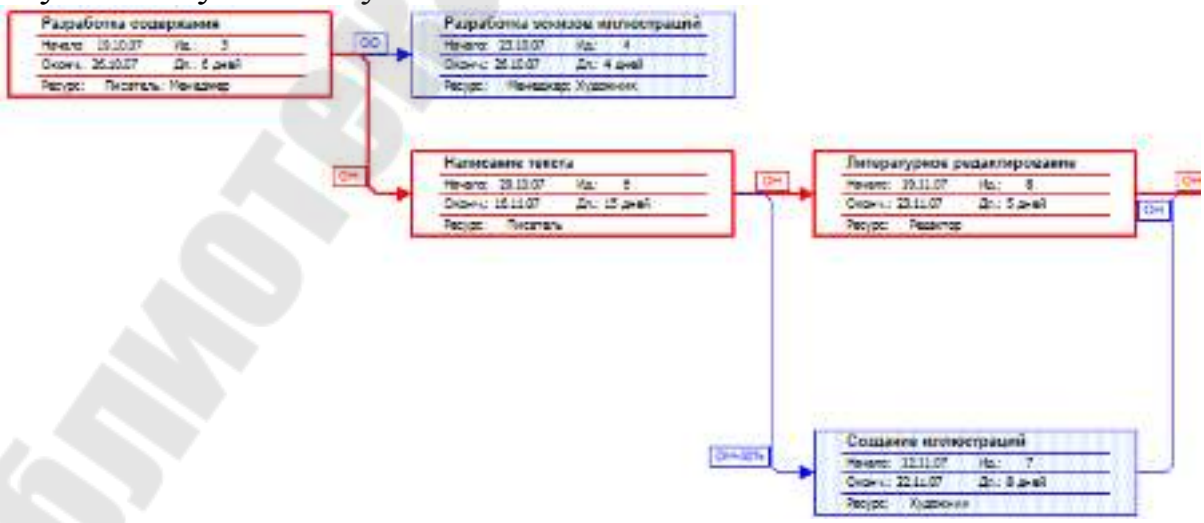


Рис. 8.2.Сетевой график

На основе сетевого графика автоматически формируется календарный план в виде диаграммы Ганта. Определив структуру расписания в виде сетевого графика, вы получаете «заготовку» календарного графика с привязкой сроков выполнения работ к реальным датам.

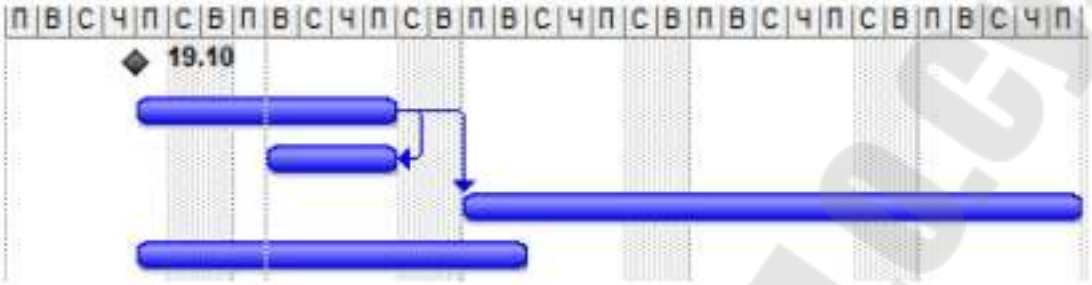


Рис. 8.3. Разные типы связей между задачами

Чтобы получить ответ на третий по счету из перечисленных выше вопросов, требуется выполнить назначение ресурсов (хотя бы на уровне текущего представления менеджера о составе и характере входящих в проект работ). В качестве ресурсов проекта могут быть заданы либо уникальные для него исполнители и материалы, либо назначены виды ресурсов, использовавшихся в предыдущих проектах (или взятые из шаблонов). Обобщенную информацию об используемых в проекте ресурсах можно получить с помощью таблицы ресурсов.

		Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь
1		Писатель	Трудовой		П	Люди	100%	0,00р./ч	0,00р./ч	7 000,00р.	По окончании	Стандартный
2		Редактор	Трудовой		Р	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорционально	Стандартный
3		Художник	Трудовой		Х	Люди	100%	70,00р./ч	140,00р./ч	0,00р.	Пропорционально	Стандартный
4		Верстальщик	Трудовой		В	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорционально	Стандартный
5		Корректор	Трудовой		К	Люди	100%	50,00р./ч	100,00р./ч	0,00р.	Пропорционально	Стандартный
6		Менеджер	Трудовой		М	Люди	100%	100,00р./ч	200,00р./ч	0,00р.	Пропорционально	Стандартный
7		Компьютер	Трудовой		К	Техника	400%	0,00р./ч	0,00р./ч	0,00р.	Пропорционально	Стандартный

Рис. 8.4.Таблица ресурсов

А более детальную — на основе анализа назначений.

Для каждого ресурса могут быть построены гистограммы его загрузки и стоимости.

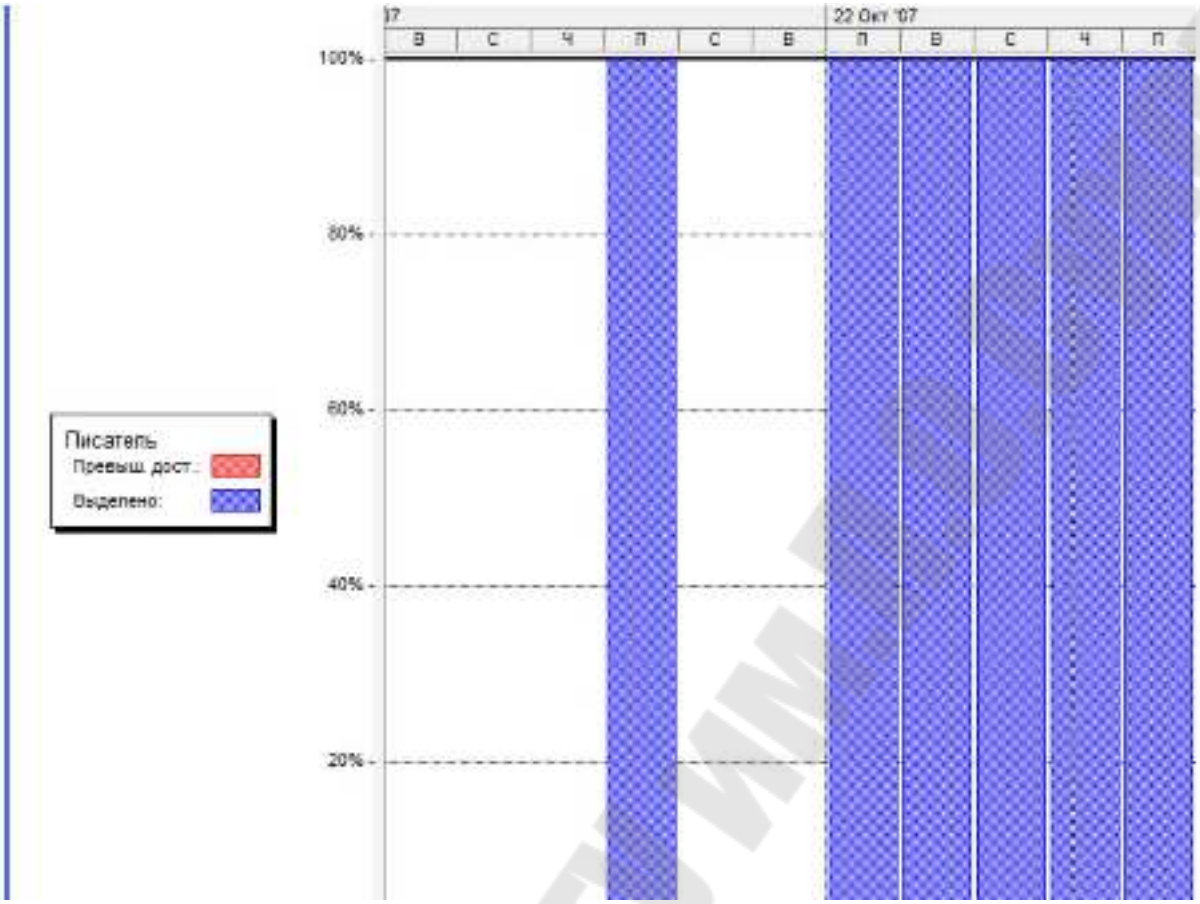


Рис. 8.5.Гистограммы работ

После назначения очередного ресурса (с указанием его стоимости и объема) выполняется **автоматический пересчет** стоимости проекта, благодаря чему очень легко получить сравнительную оценку различных вариантов назначений.

Для проведения стоимостного анализа проекта в MS Project 2007 используется так называемый «метод освоенного объема» (Earned Value Analysis), с помощью которого может быть проведен анализ затрат либо на текущую дату, либо на заданную календарную дату.

Любой, даже самый хороший план не застрахован от случайностей.

Чтобы адекватно анализировать риски проекта, необходимо иметь его детализированный план. Так что самое лучшее время, чтобы выполнить начальный анализ рисков, — непосредственно перед сохранением базового плана. Как правило, выбор методов и средств для анализа рисков зависит от специфики проекта, состава и уровня подготовки группы проекта. В частности, простым и вместе с тем эффективным средством является сравнение нескольких версий (сценариев) расписания проекта.

При использовании **анализа по** методу **PERT** таких сценариев должно быть три: **наиболее вероятный (ожидаемый), оптимистичный и пессимистичный**. Для сравнительной оценки длительности проекта по этим трем сценариям в составе MS Project 2007 имеется специальный инструмент — процедура *PERT,* С се помощью вы можете описать и сравнить между собой параметры проекта для трех альтернативных сценариев развития событий.

Для более сложных ситуаций могут быть созданы соответствующие макросы, реализованные с помощью VBA (язык программирования Visual Basic Application).

Завершая короткий обзор основных возможностей MS Project 2007, отметим, что на любой стадии работы над проектом вы всегда будете чувствовать поддержку со стороны разработчиков. Либо в виде всплывающих окон с подсказками, либо в форме смарт-тегов, либо в какой-то другой форме. Например, если вы станете последовательно вводить одинаковые значения параметров для нескольких задач, то на экране появится окно *Мастера планирования* с подсказкой, как избежать повторного ввода.

При запуске Microsoft Project выглядит так.

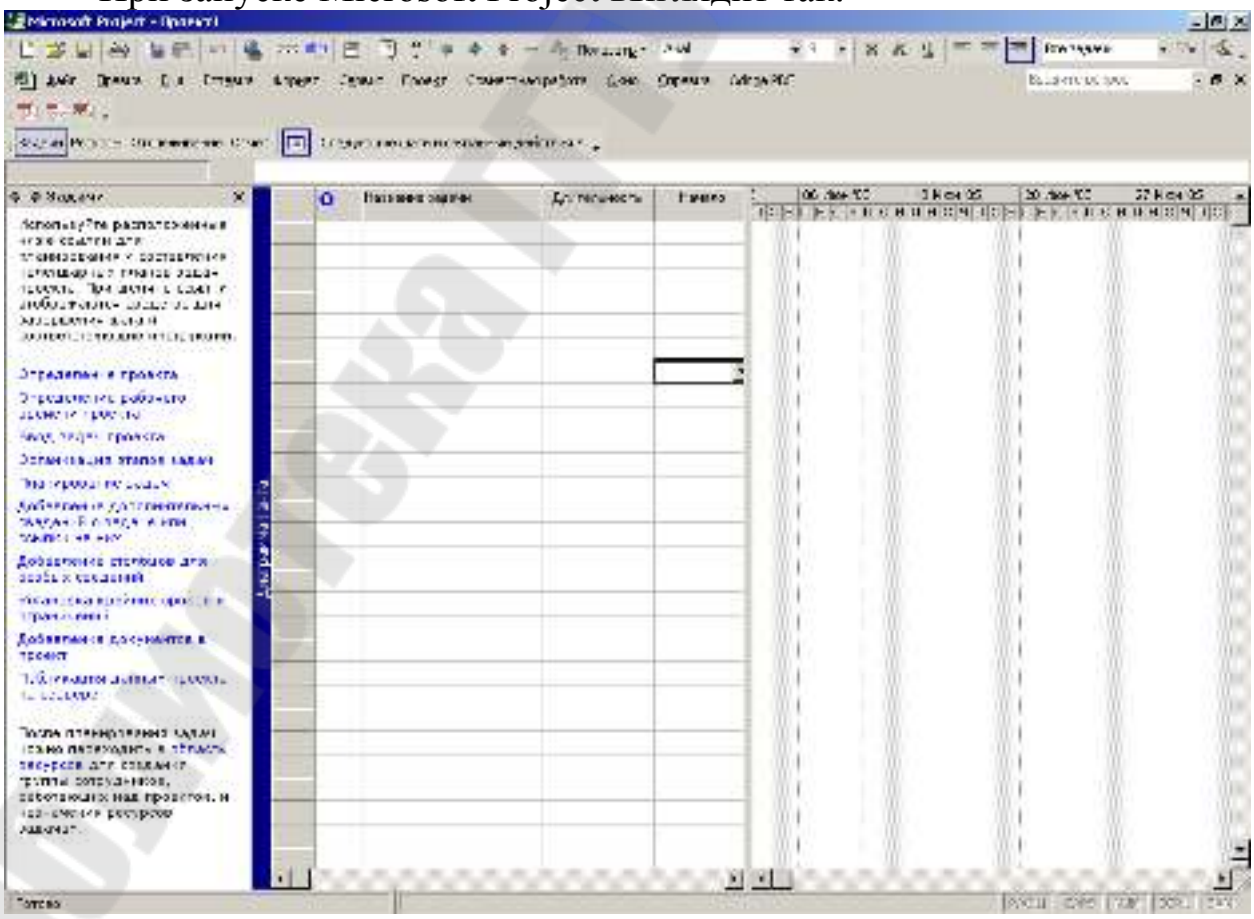


Рис. 8.6.Главное окно MS Project

Сверху расположена строка меню, под ней – панель инструментов, ниже – рабочая область программы.

Среди панелей инструментов есть особая панель *Консультант* (*Project Guide*), которая расположена над рабочей областью программы. Названия кнопок этой панели соответствуют основным объектам, с которыми может работать MS Project: *Задачи (Tasks)*, *Ресурсы (Resources)*, *Отслеживание (Track)*, *Отчет (Report)*. При щелчке по любой из этих кнопок на панели в левой части рабочей области, которая называется *Областью задач* (*Task Pane*) отображается список возможных действий с выбранным объектом.

Настройка параметров программы

Перед началом работы MS Project необходимо настроить некоторые параметры программы. Для перехода к настройке нужно выбрать команду Сервис → Параметры (Tools → Options). На экране появится диалоговое окно настроек с несколькими закладками, на которых сгруппированы параметры, определяющие работу программы. Почти на каждой закладке есть кнопка *По умолчанию* (*Set as default*), позволяющая сохранить сделанные настройки так, чтобы они автоматически применялись во всех последующих проектах.

На рисунке 8.7 представлена закладка *Вид* (*View*).

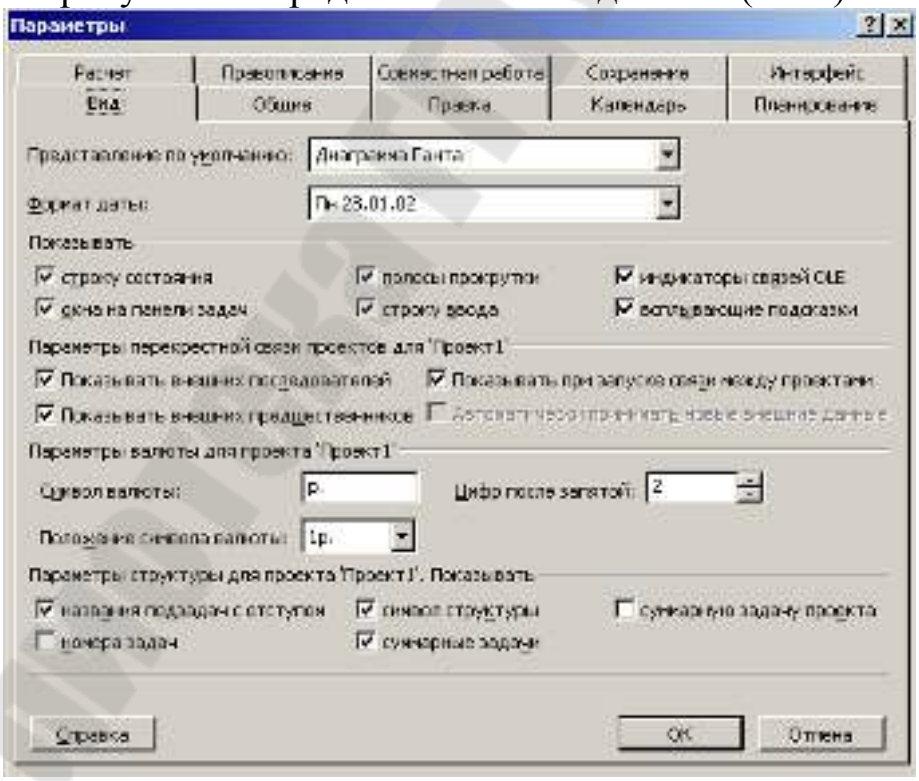


Рис. 8.7. Параметры MS Project. Закладка «Вид»

Прежде чем начать составление плана проекта необходимо ввести в диалоговом окне начальные данные о проекте. Флажок *Запрос на ввод сведений о проекте для новых проектов (Prompt for project info for new projects)* обеспечивает автоматический вывод на экран необходимого диалогового окна при создании нового проекта.

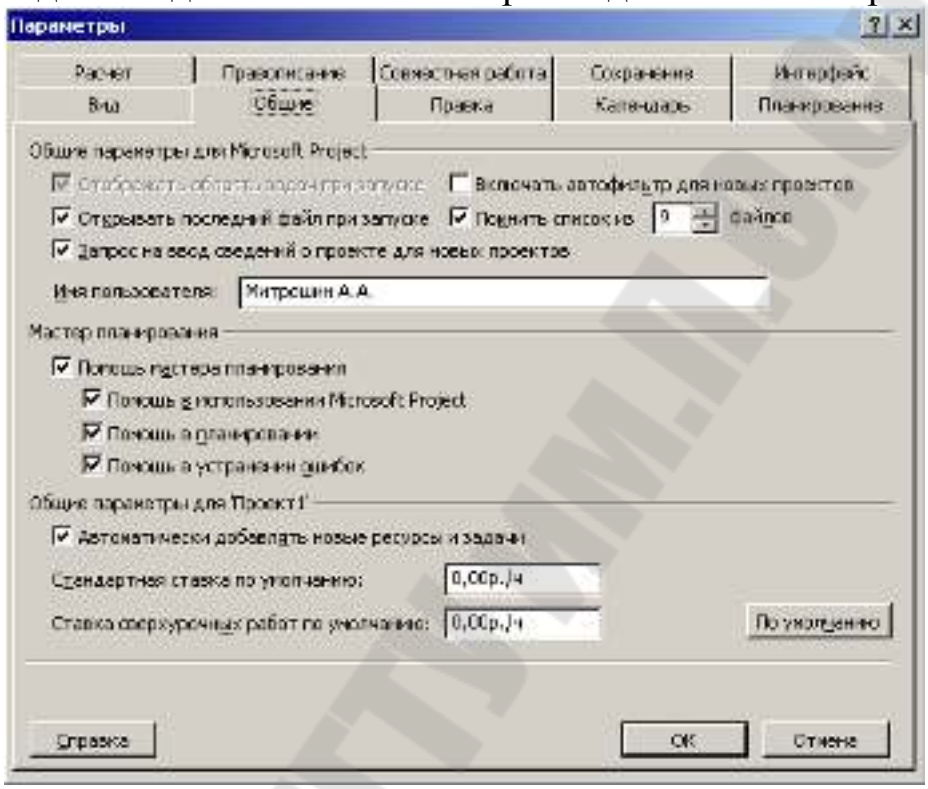


Рис. 8.8. Параметры MS Project. Закладка «Общие»

В MS Project имеется особый режим просмотра таблиц – Автофильтр (AutoFilter), при котором в заголовке каждой таблицы отображается кнопка для ее сортировки. Поскольку этот режим требуется не всегда и его можно включить кнопкой *Форматирование (Formatting)* панели инструментов, флажок *Включать автофильтр для новых проектов (Set AutoFilter for new projects)* удобнее сбросить.

В последнем блоке настроек нужно установить флажок *Автоматически добавлять новые ресурсы и задачи (Automatically add new resources and tasks)*. Это обеспечит удобный ввод данных в проект.

Настройки редактирования помещены в закладке *Правка(Edit)* (рисунок 8.9) и сгруппированы в двух разделах (вверху закладки), один из которых содержит элементы настройки, относящиеся к программе целиком, а второй (внизу закладки) к открытому в данный момент проекту.

Флажок *Перетаскивание ячеек (Allow cell drag and drop)* определяет, будет ли можно перетаскивать ячейки таблиц с помощью мыши, а флажок *Переход к следующему полю после ввода (Move selection after enter)* – будет ли перемещаться курсор в следующую ячейку, после того как при редактировании текущей ячейки нажата клавиша Enter. Флажок *Правка прямо в ячейке(Edit directly to cell)* определяет, можно ли редактировать данные непосредственно в ячейке таблицы. Если этот флажок сбросить, то для редактирования данных будет необходимо выделить ячейку и вводить ее содержимое в строку ввода.

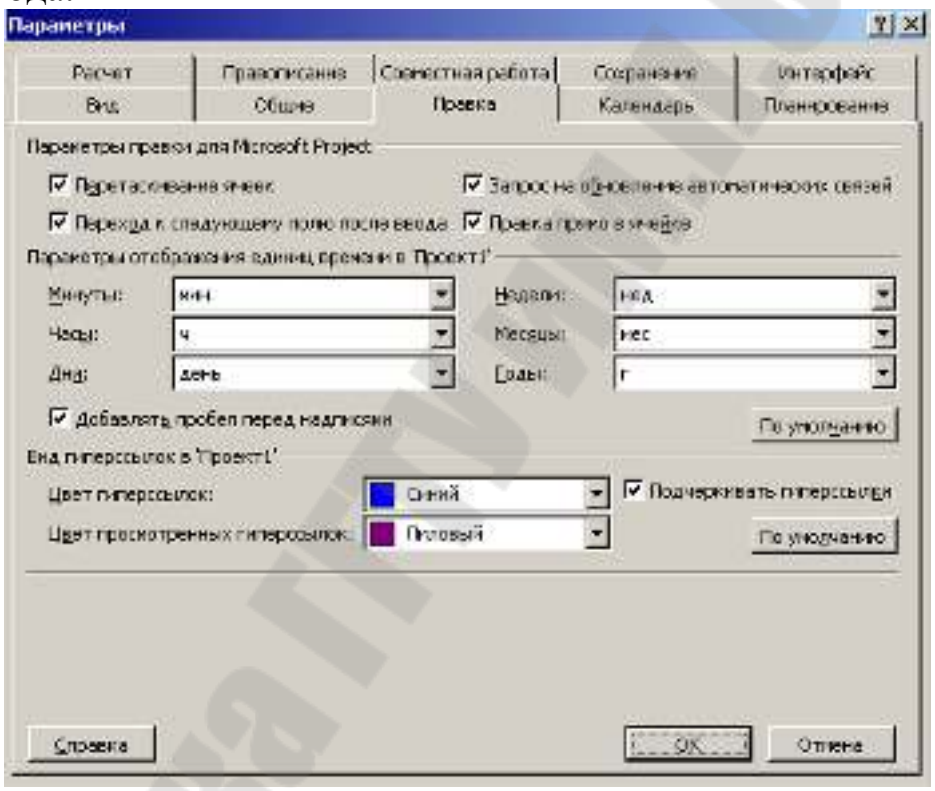


Рис. 8.9. Параметры MS Project. Закладка «Правка»

Флажок *Запрос на обновление автоматических связей (Ask to update automatic links)* управляет способом обновления объектов из других файлов, внедренных в файл проекта. Обновление может происходить автоматически или по запросу. Для автоматического обновления внедренных объектов флажок необходимо сбросить.

От состояния флажка *Добавлять пробел перед надписями (Add space before label)* зависит, будет ли вставляться пробел между количеством единиц и их обозначением. По умолчанию флажок установлен, и сбрасывать его не желательно.

Закладка *Календарь* (рисунок 8.10) используется для ввода, просмотра и изменения параметров даты и времени. Все эти

параметры являются локальными и сохраняются вместе с текущим проектом. Имеются следующие возможности.

- Задание первого дня недели и первого месяца финансового года для данного проекта.
- Задание времени начала и окончания по умолчанию для тех ограничений задач, в которых вводится дата, но не вводится время.
- Задание количества часов в дне или неделе, а также количества дней в месяце для ввода значений длительности и трудозатрат.

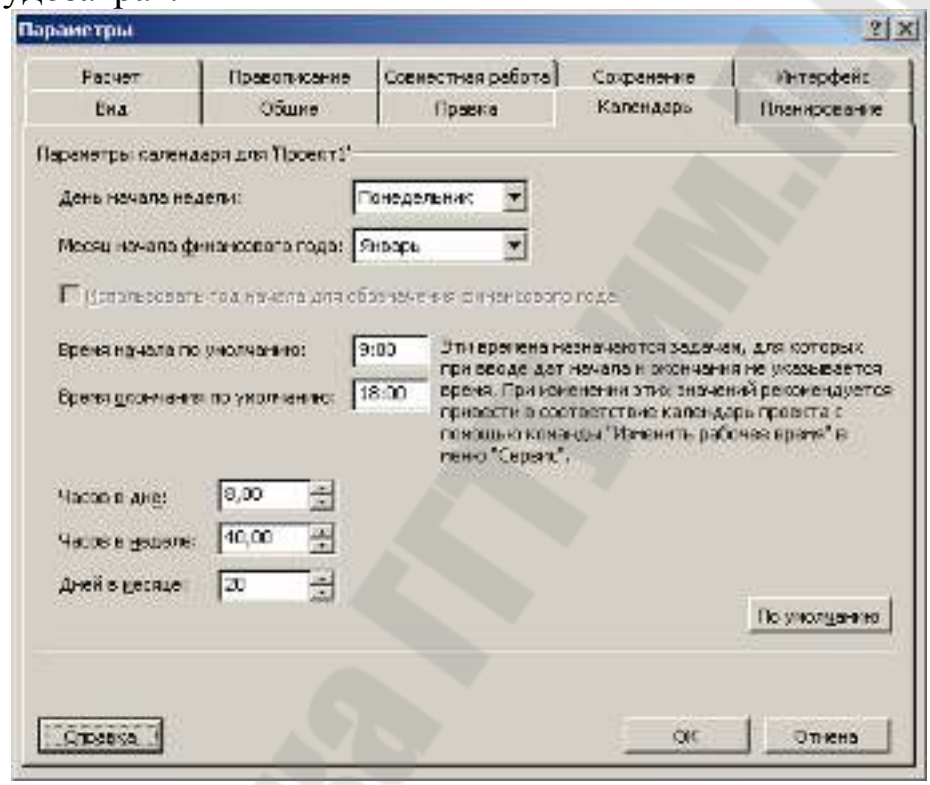


Рис. 8.10. Параметры MS Project. Закладка «Календарь»

Изменение параметров данного диалогового окна не влияет ни на календарь проекта, ни на календари рабочего времени ресурсов. Оно влияет только на преобразование длительностей в соответствующие временные интервалы. Заданный на этой вкладке временной интервал применяется также к преобразованию длительности в значения трудозатрат. Подобным же образом осуществляется преобразование временных значений полей Часов в неделе и Дней в месяце.

Закладка *Планирование* используется для ввода, просмотра и изменения параметров планирования задач. Возможны как установка глобальных параметров планирования для программы в целом, так и установка локальных параметров планирования для текущего

проекта, включая даты начала новых задач, единицы измерения времени, используемые для длительности и трудозатрат, и стандартный тип задачи.

Длительность вводится в - этот параметр определяет единицу измерения времени, используемую по умолчанию в поле «Длительность». Программа будет вводить ее автоматически, если пользователь при вводе значения в поле «Длительность» не укажет единицу измерения времени. Если в поле «Длительность» ввести другую единицу измерения времени, программа будет автоматически вводить новую единицу, если значение времени будет введено без указания единицы измерения.

Тип задач по умолчанию - определяет влияние изменения существующих сведений о назначениях — трудозатрат, единиц назначения или длительности — на расчет двух других полей назначений для данной задачи. По умолчанию он задает тип для всех новых задач.

Существуют следующие типы задач.

Фиксированная длительность. Этот тип задачи выбирается в случае, если требуется сохранить постоянной длительность задачи независимо от любых изменений единиц назначения или трудозатрат. В задачах с фиксированной длительностью выполняются следующие действия.

- При изменении единиц заново вычисляет трудозатраты.
- При изменении длительности в задаче с фиксированной длительностью заново вычисляются трудозатраты.
- При изменении количества трудозатрат заново вычисляются единицы.

Фиксированные единицы. Тип задачи Фиксированные единицы выбирается в случае, если требуется сохранить постоянным число единиц назначения независимо от каких-либо изменений длительности или трудозатрат. Этот параметр применяется по умолчанию. В задачах с фиксированными единицами выполняются следующие действия.

- При изменении единиц в задаче с фиксированными единицами заново вычисляется длительность.
- При изменении длительности заново вычисляются трудозатраты.
- При изменении количества трудозатрат заново вычисляется длительность.

Фиксированные трудозатраты. Этот тип задачи выбирается в случае, если требуется сохранить постоянным количество трудозатрат независимо от каких-либо изменений длительности или единиц назначения ресурсов для задачи. Поскольку задачи с фиксированными трудозатратами по определению являются задачами с фиксированным объемом работ, флажок *Фиксированный объем работ* устанавливается для них автоматически. В задачах с фиксированными трудозатратами выполняются следующие действия:

- при изменении единиц заново вычисляется длительность;
- при изменении длительности заново вычисляются единицы;
- при изменении количества трудозатрат для задач с фиксированными трудозатратами производится перерасчет длительности.

Флажок *Показывать сообщения о планировании (Show scheduling messages)* определяет, будет ли MS Project сообщать о несоответствиях в плане проекта.

Таблицы

Таблицы являются основным средством хранения данных в MS Project.

В проектном файле все данные хранятся в виде двух таблиц, одна из которых содержит информацию о задачах, а вторая – о ресурсах проекта, то есть задействованных при выполнении проекта людях и материальных ценностях. Две этих внутренних таблицы состоят из множества полей, большинство из которых не используется. Отображаются только столбцы, включенные в определенные *представления*.

В MS Project заложен набор предопределенных таблиц, каждая из которых содержит несколько полей одной из внутренних таблиц проектного файла. Списки предопределенных таблиц MS Project с информацией о задачах и ресурсах приведены в табл. 8.1 и 8.2.

Таблица 8.1

Таблица	***Содержание***
Исходный план (Baseline)	Данные из базового плана проекта
Даты ограничений (Constraint Dates)	Ограничения задачи (даты ограничений и типы)
Затраты (Cost)	Стоимость задач и проекта
Задержка (Delay)	Информация для выравнивания загрузки ресурсов

Продолжение табл. 8.1

Освоенный объем (Earned Value)	Общая таблица для сравнения запланированного и фактического объемов работ, запланированной и фактической стоимости
Индикаторы календарного плана освоенного объема (Earned Value Schedule Indicators)	Сравнение запланированного и фактического графиков выполнения проекта
Показатели затрат для освоенного объема (Earned Value Cost Indicators)	Сравнение запланированного и фактического проектного бюджета
Ввод (Enter)	Таблица для ввода общей информации о задаче
Экспорт (Export)	Информация для экспорта данных о задачах во внешний файл
Гиперссылка (Hyperlink)	Связанные с задачей ссылки, ведущие на внешние сайты
Ожидаемый сценарий (PA_Expected Case) **Оптимистичный сценарий (PA_Optimistic Case)** **Пессимистичный сценарий (PA_Pessimistic Case)**	Три таблицы для анализа проекта по методике PERT. Идентичны по структуре и содержат данные об ожидаемом, оптимистичном и пессимистичном планах проекта
Ввод для анализа PERT (PA_PERT Entry Sheet)	Таблица для ввода данных, используемых при проведении анализа плана по методике PERT
Сводные задачи (Rollup Table)	Служит для оптимизации отображения сводных задач
Schedule (календарный план)	Расписание начала и окончания выполнения задач, интервалы между задачами
Суммарные (Summary)	Общая информация о задачах проекта
Отслеживание (Tracking)	Информация о ходе выполнения проекта

Окончание табл. 8.1

Использование (Usage)	Задачи проекта, объемы работ, длительность, даты начала и окончания
Расхождение (Variance)	Данные об отклонении от плана при выполнении работ
Трудозатраты (Work)	Информация об объеме работы по проекту

Таблица 8.2

Таблица с информацией о ресурсах

Таблица	***Содержание***
Стоимость (Cost)	Стоимость ресурсов проекта
Приобретенная стоимость (Earned Value)	Сравнение запланированной и фактической стоимостей ресурса в проекте
Ввод (Entry)	Таблица ввода общей информации о ресурсе
Ввод материального ресурса (Entry – Material Resources)	Таблица для ввода общей информации о материальном ресурсе
Ввод нематериального ресурса (Entry – Work Resources)	Таблица для ввода общей информации о нематериальном ресурсе
Экспорт (Export)	Информация для экспорта данных о ресурсах во внешний файл
Гиперссылка (Hyperlink)	Связанные с ресурсом ссылки, ведущие на внешние сайты в Internet и Intranet
Сводная (Summary)	Общая информация о ресурсах проекта
Использование (Usage)	Информация о работе, на которую выделены ресурсы
Работа (Work)	Информация об объеме работы, на которую выделены ресурсы

Таблицы отображаются в *представлениях*, причем есть представления, в которых таблицы совмещены с диаграммой, например диаграммой Ганта (Gantt Chart). Существуют и представления, состоящие только из таблицы. Какая из таблиц загружается по умолчанию при загрузке программы, определяется в настройках.

Переключение между таблицами осуществляется с помощью команды Вид→Таблица (View→Table). В пункте меню Вид→Таблица перечислены наиболее часто используемые таблицы для текущего представления. Если в этом пункте меню нет нужной таблицы, то ее можно найти в пункте *Другие таблицы* (*More tables*).

8.2. Подготовка и составление плана в Microsoft Project

Управление проектом заключается в составлении плана и отслеживании хода работ по нему. Проекты могут осуществляться в любой области деятельности. Так проектом может быть разработка информационной системы, выполнение курсового или дипломного проекта, постройка здания, проведение предвыборной компании и т.д.

Проект предпринимается для достижения определенного результата в определенные сроки и за определенные деньги. План проекта составляется для того, чтобы определить, с помощью каких работ будет достигаться результат проекта, какие люди и оборудование будут нужны для выполнения работ, в какое время эти люди и оборудование будут заняты работой по проекту. Поэтому проектный план содержит три основных элемента: задачи (tasks), ресурсы (resource) и назначения (assignment).

Задачей называется работа, осуществляемая в рамках проекта для достижения определенного результата. Поскольку проект обычно содержит много задач, то для удобства отслеживания плана их объединяют в группы, или **фазы**. Совокупность фаз проекта называется его жизненным циклом.

Фаза проекта состоит из одной или нескольких задач, в результате выполнения которых достигается один или несколько основных результатов проекта. Если для достижения результатов задачи нужно выполнить только ее, то для достижения результата фазы нужно выполнить группу других задач. В этом заключается отличие фазы от задачи – ее результат суммирует результаты других задач. Поэтому в MS Project фазы называются **суммарными**

задачами (summary task). Фазы могут состоять как из задач, так и других фаз.

Проект разбивается на фазы для удобства контроля хода работы. По завершении проектной фазы обычно осуществляется анализ полученных результатов, чтобы с минимальными затратами определить и исправить ошибки.

Задачи, в результате выполнения которых достигаются промежуточные цели, называются **завершающими задачами**. В MS Project они называются **вехами** (milestone). Обычно результатом фазы является достижение некоторой промежуточной цели, поэтому вехой в плане проекта принято обозначать последнюю задачу фазы, в результате которой достигается ее результат. Если такой задачи нет, а фазовый результат достигается, например, одновременным завершением нескольких задач, то создается фиктивная завершающая задача, длительность которой устанавливается равной 0 дней, и на нее не выделяются исполнители. Она присутствует в плане исключительно для обозначения момента завершения фазы, что облегчает отслеживание плана проекта.

Длительность задачи - это период рабочего времени, необходимый для выполнения задачи. Длительность задачи может не соответствовать **трудозатратам** занимающегося задачей сотрудника. Длительность (duration) соответствует времени, через которое будет получен результат работы, а трудозатраты (work) – времени, затраченному сотрудниками на получение результата.

Задачи в плане проекта взаимосвязаны, например, часто одна задача не может начаться до тех пор, пока не будет закончена другая. На плане проекта **зависимости** (dependencies) обозначаются с помощью **связей** (links). Оба этих термина – **зависимость** и **связь** - используются с одним и тем же смыслом, обозначая логику, определяющую последовательность выполнения работ в плане проекта.

Под **ресурсами** понимаются сотрудники и оборудование, необходимые для выполнения проектных задач. Каждый сотрудник, участвующий в проекте, получает определенную **роль**, соответствующую его квалификации. При составлении списка ресурсов часто используется ролевое планирование. Например, сначала определяется, что для исполнения работ требуются три программиста и один менеджер, а затем, когда план проекта утвержден, подбираются конкретные сотрудники для этих ролей.

Важное свойство ресурсов – **стоимость** (cost) их использования в проекте. В MS Project существуют два типа стоимости ресурсов: повременная ставка (rate) и стоимость за использование (cost per use). Повременная ставка выражается в стоимости использования ресурса за единицу времени. Обычно почасовая ставка используется для учета стоимости не материальных ресурсов. Величина затрат на использование обозначает стоимость использования оборудования или сотрудника в задаче, которая не зависит от того, сколько времени задействован в задаче сотрудник или материальный ресурс. Общие затраты на использование такого ресурса определяются путем умножения стоимости использования на число задач, в которых он задействован. У ресурса может быть указана стоимость как одного из двух типов, так и обоих.

Назначения – это связь определенной задачи и ресурсов, необходимых для ее выполнения. На одну задачу может быть назначено несколько ресурсов, причем как материальных, так и нематериальных. Назначения объединяют в плане задачи и ресурсы, делая план целостным. Благодаря назначениям решается целый ряд задач планирования:

- определяются ответственные лица за исполнение задач;
- когда определены задачи, за которые отвечает ресурс, можно рассчитать общий объем времени, затрачиваемый им на проект, а значит и его стоимость;
- определив стоимость участия всех ресурсов в проекте, можно определить общую стоимость проекта;
- назначая ресурсы на задачи, можно сокращать срок выполнения работ, выделяя на них больше ресурсов, и тем самым, сокращая общую длительность проекта.

Большинство проектов имеют определенную дату окончания, бюджет и объем работ.

Тройку «время», «деньги», «объем работ» часто называют **проектным треугольником** потому, что при внесении изменений в один из этих элементов, меняются оба другие. Хотя для проекта в равной степени важны все три элемента, **один из них**, как правило, **имеет наибольшее влияние** на другие в зависимости от выбранного приоритета. Например, если изменить план проекта, укоротив **расписание**, то либо возрастает стоимость проекта, либо уменьшается объем выполненных работ. Если изменить план проекта с целью **уменьшения его бюджета**, то может возрасти длительность

выполнения проекта и уменьшиться объем работ. Если **увеличить объем работ**, то проект будет длиться дольше и стоить дороже. В общем случае изменения в плане зависят от специфики проекта. В некоторых случаях сокращение времени увеличивает стоимость, а в других – уменьшает ее.

Качество – четвертый элемент проектного треугольника. Изменения, вносимые в любую из сторон треугольника, практически всегда влияют на качество. Качество не является стороной треугольника – это результат действий со временем, стоимостью и объемом работ. Например, если существует лишнее время в расписании, то можно увеличить объем работ, добавив новые задачи и, возможно, увеличив длительность проекта. С этими дополнительными задачами и временем можно добиться более высокого качества выполнения проекта. С уменьшением объемов работ у проекта будет меньше шансов выйти на требуемый уровень качества, поэтому снижение расходов может привести к ухудшению качества проекта.

Составление плана проекта в общем виде заключается в описании задач проекта, доступных ресурсов и определении взаимосвязей между ними с помощью назначений. Однако, при составлении расписания работ в MS Project, количество операций несколько увеличивается.

Планирование начинается с определения проекта, то есть описания его ключевых характеристик. Затем составляется список фаз и задач, а также список необходимых для их выполнения ресурсов. После этого в план вносится дополнительная информация о задачах и ресурсах, которая будет использоваться при определении назначений и в дальнейшем при проведении работ по плану (***отслеживание работ***). Далее осуществляются назначения, после чего проект оптимизируется, если его длительность или бюджет окажутся больше ожидаемых.

Составление плана невозможно без задания ключевых параметров проекта, таких как его длительность, рабочее время и методика планирования.

Чтобы составить расписание (план) работ в MS Project нужно создать файл нового проекта, щелкну по кнопке *Создать* (*New*) на панели инструментов. Если установлен соответствующий флажок в настройках программы, то откроется диалоговое окно *Сведения о проекте (Project Information)* (рис. 8.11). Чтобы изменить параметры

проекта в дальнейшем, это диалоговое окно можно вызвать командой *Проект→ Сведения о проекте (Project→Project Information)*.

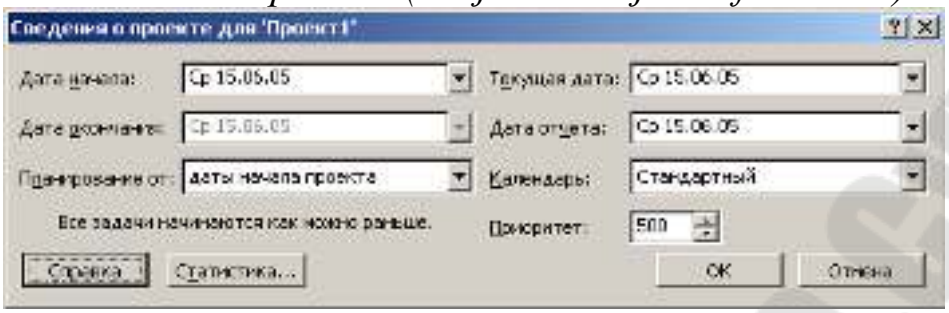


Рис. 8.11. Окно сведений о проекте

Проект можно планировать двумя способами: от даты начала проекта или от даты окончания. Если у проекта нет жесткой даты окончания, то при планировании применяется первый способ: фиксируется дата, когда необходимо начать выполнение проекта, и в ходе составления плана определяется дата его завершения. Если же проект должен быть завершен к фиксированной дате, то используется противоположный способ: фиксируется дата окончания и в ходе составления плана определяется, когда выполнение проекта должно быть начато, чтобы все работы были выполнены в срок.

Способ планирования выбирается в раскрывающемся списке *Планирование от (Schedule from)*.

Даты начала и окончания проекта выбираются в списках *Дата начала (Start date)* или *Дата окончания(Finish date)*. Зафиксировать можно только одну из дат в соответствии с выбранным способом планирования.

Значения текущей даты и даты отсчета выбираются в раскрывающихся списках *Текущая дата (Current date)* и *Дата отчета (Status date)*. По умолчанию текущая дата соответствует системной дате операционной системы, а дата отсчета равна текущей дате и поэтому в окне сведений о проекте в качестве ее значения выбрано НД (NA). Любое из значений можно изменить независимо от другого. При отслеживании проекта и вводе сводной информации используется дата отсчета, а не текущая дата.

По умолчанию программа считает, что все выполненные трудозатраты относятся ко времени до даты отсчета, а все оставшиеся трудозатраты – ко времени после даты отчета. Если выполнение задачи запланировано после даты отсчета, но вносятся данные о фактических трудозатратах по этой задаче, то MS Project изменит выполненную часть задачи так, чтобы она закончилась к дате отсчета, а выполнение оставшихся трудозатрат остается на будущее.

Чтобы определить рабочее время, в рамках которого будут выполняться работы в раскрывающемся списке *Календарь (Calendar)* нужно выбрать один из доступных календарей. Календарем в MS Project называется набор параметров, определяющих перечень рабочих и нерабочих дней, а также рабочее время в каждом из рабочих дней. В стандартной поставке в этом списке присутствуют три пункта:

1) стандартный (Standard);

2) 24 часа (24 Hours);

3) ночная смена (Night shift).

Первый календарь соответствует стандартному расписанию с 8 часовым рабочим днем, второй – круглосуточному рабочему дню, а третий предполагает круглосуточный режим работы с перерывами.

Очень часто входящие в состав поставки календари не подходят для проекта. В этом случае можно изменить существующий календарь или создать новый. В MS Project существует возможность создавать как групповые (или базовые), так и личные календари. Последние отражают персональные расписания отдельных сотрудников. Поэтому при создании базового календаря в него следует вносить только настройки, общие для всех участников проекта или группы, к которой относится календарь. Специфические настройки заносятся в личный календарь каждого сотрудника.

В поле *Приоритет* указывается число в диапазоне от 0 до 1000, которое используется при выравнивании загрузки ресурсов между разными проектами. Чем больше число, тем выше приоритет проекта.

После нажатия кнопки *Ок* создается новый файл проекта.

Первым шагом в планировании является составление списка задач. Только после этого можно оценить сроки и стоимость выполнения каждой задачи.

Определение состава проектных работ начинается с определения этапов (или фаз) проекта. После того, как состав фаз и результаты определены, нужно определить последовательность фаз относительно друг друга и крайние сроки их выполнения. Затем нужно выяснить, из каких работ состоят фазы, в какой последовательности выполняются работы внутри фаз и в какие сроки необходимо уложиться при выполнении каждой работы.

Определять состав работ удобно поэтапно. Сначала создается *скелет плана работ*, состоящий из фаз, их результатов и нескольких основных задач. Затем в план добавляются остальные задачи,

определяются их длительности и связи. Затем определяются ключевые даты проекта, которые включают крайние сроки достижения результатов проекта и некоторые другие ограничения по времени. Наконец, в план добавляется дополнительная информация о задачах.

Создадим новый проект с методикой планирования от даты начала. Используем стандартный календарь. В качестве даты проекта выберем предлагаемую по умолчанию.

План работ удобнее всего составлять в представлении *Диаграмма Ганта (Gantt Chart)*. Диаграмма Ганта [2] представляет собой хронограмму на полный набор работ, которая широко используется в настоящее время. Для добавления задачи в план проекта нужно установит курсор в таблицу слева от диаграммы и ввести название задачи в поле *Название задачи (Task name)*. После этого отрезок, символизирующий задачу, появится на диаграмме.

Добавление в план фазы не отличается от добавления задачи – любая задача автоматически становится фазой, как только у нее появляется вложенная задача, то есть задача, находящаяся на следующем уровне структуры плана. До тех пор пока у задачи нет вложенных задач, она не является фазой.

Чтобы поместить задачу на более низкий уровень структуры, нужно установить курсор в строку с задачей и на панели инструментов щелкнуть по кнопке *На уровень ниже* (⇒) в панели инструментов или нажать комбинацию клавиш Alt+Shift+→. Для перемещения задачи на более высокий уровень структуры нужно щелкнуть на кнопке *На уровень выше* (⇐) панели инструментов или использовать комбинацию клавиш Alt+Shift+←.

Пример создания скелетного плана приведен на рисунке 8.12.

Рис. 8.12 Пример создания скелетного плана

Из рисунка 8 видна разница графического отображения фазы «Выполнение курсового проекта» от задачи «Получит задание на выполнение курсового проекта».

Результаты фаз вводятся в виде завершающих задач, и эти задачи могут не обозначать реальной деятельности. Например, результатом фазы «Выполнение курсового проекта» является

сданный курсовой проект. Для того, чтобы указать тот факт, что данная задача является завершающей, ее длительность устанавливается равной 0 (рис. 8.13).

Рис. 8.13. Добавление завершающей задачи

Добавим теперь завершающие работы к фазам «Получить задание на выполнение курсового проекта», «Выполнить курсовой проект», «Защитить курсовой проект». Сейчас это не фазы, но они станут фазами, если к этим работам добавить подчиненные работы (рис. 8.14).

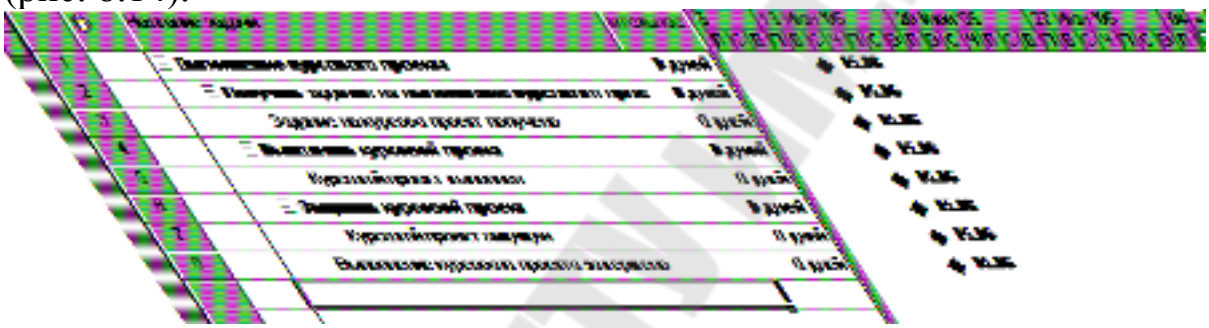


Рис. 8.14. Добавление завершающей задачи к каждой фазе

Поскольку сейчас каждая фаза содержит только завершающую задачу, то есть задачу длительности 0, а длительность выполнения фазы определяется длительностями входящих в фазу задач, то длительность каждой фазы равно 0. Поэтому фазы помечены как завершающие задачи. Добавим в фазы задачи, решением которых достигается цель фазы.

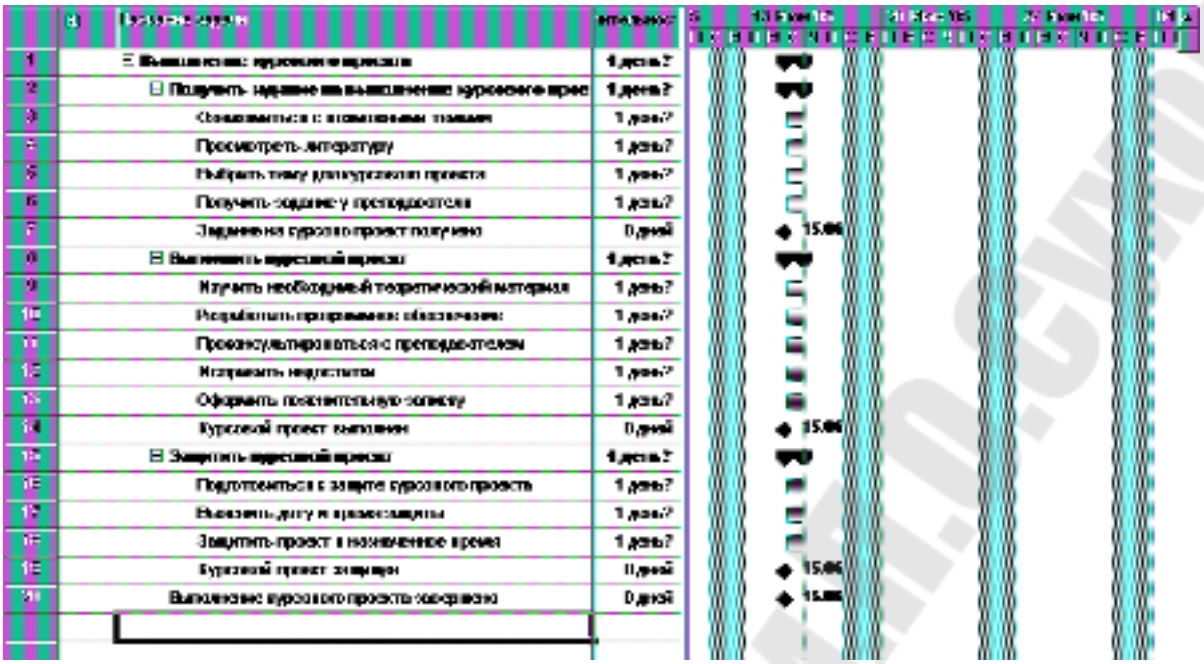


Рис. 8.15. Концевые фазы

После определения состава работ можно переходить к определению длительностей задач и связей между ними.

Длительность задач определяется значением, вводимым в поле *Длительность (Duration)*. Длительность фаз вводить нельзя – она рассчитывается автоматически.

При создании задач, им автоматически присваивается длительность 1 день. После единицы измерения времени добавляется вопросительный знак (?). Этот знак означает, что указанная длительность является приблизительной (estimated) и требует уточнения в дальнейшем. После того, как значение длительности задачи будет отредактировано, вопросительный знак исчезнет. При желании вопросительный знак можно поставить и самостоятельно, чтобы отметить тот факт, что длительность помеченной таким образом задачи, должна быть скорректирована (рис. 8.16).

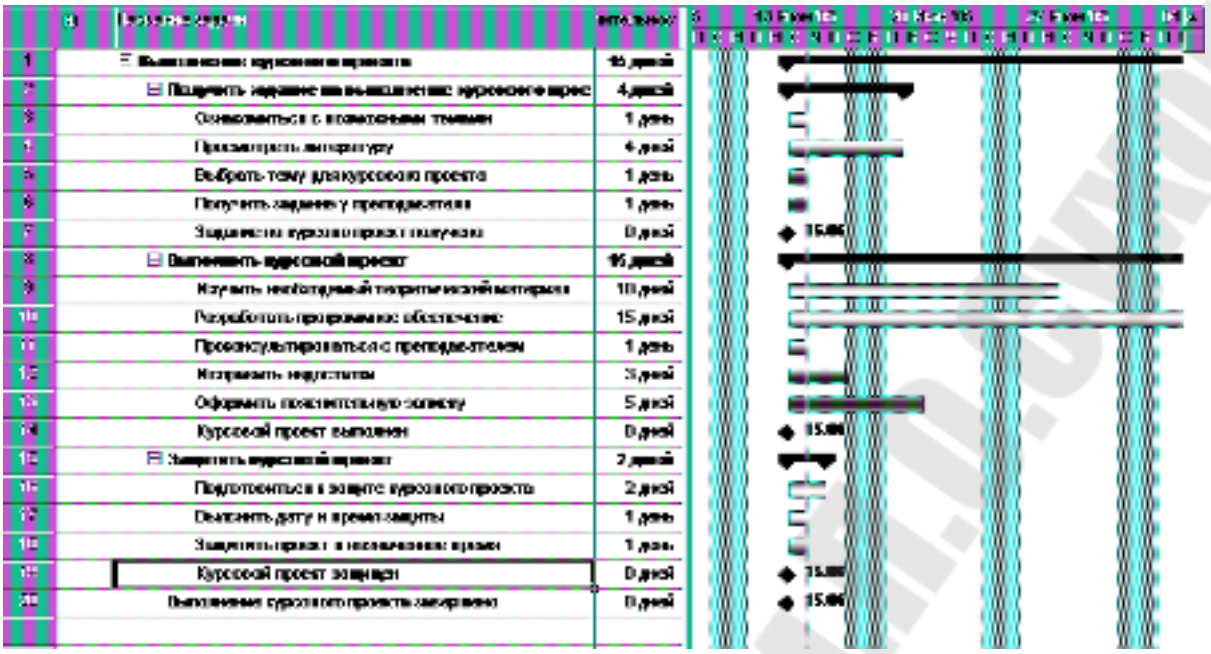


Рис. 8.16. Задачи с добавленными длительностями

После определения длительностей задач, можно переходить к определению зависимостей между задачами. Например, разработка программного обеспечения по проекту невозможна до тех пор, пока не будет определена тема работы и не будет изучен соответствующий теоретический материал.

Задача, влияющая на другую задачу, называется *предшественником (Predecessor)*, а задача, зависящая от другой, называется *последователем (Successor)*. Например, задача «Изучить необходимый теоретический материал» является предшественником для задачи «Разработать необходимое программное обеспечение», а задач «Разработать необходимое программное обеспечение» последователь для задачи «Изучить необходимый теоретический материал».

Каждая задача может иметь неограниченное количество предшествующих и последующих задач. Связи могут объединять и фазы. Все принципы организации связей между задачами применимы и к фазам.

В MS Project имеется четыре типа связей между задачами.

1) Связь типа *«Окончание – начало » (Finish to Start)*, или сокращенно *ОН (FS)*. Это наиболее распространенный тип связи между задачами, при котором задача В не может начаться раньше, чем закончиться задача А. Графически этот тип связи описывается следующим образом:

2) Связь типа *«Начало – Начало» (Start to Start)*, или сокращенно *НН (SS)*, обозначает зависимость, при которой задача В не может начаться до тех пор, пока не началась задача А. С помощью такой связи обычно объединяются задачи, которые должны выполняться почти параллельно.

Графически этот тип связи описывается следующим образом:

3) Связь типа *«Окончание-Окончание» (Finish to Finish)*, или сокращенно *ОО (FF)* обозначает зависимость, при которой задача В не может закончиться до тех пор, пока не закончилась задача А. Обычно такой связью объединяются задачи, которые должны выполняться почти одновременно, но при этом одна не может закончиться, пока не завершена другая. Например, сдача программы идет одновременно с исправлением ошибок, и пока исправление ошибок не завершено, сдача программы не может завершиться.

4) Связь типа *«Начало - Окончание» (Start to Finish)*, или сокращенно *НО (SF)*. Обычно такая связь используется в том случае, когда А является задачей с фиксированной датой начала, а задача В не может закончиться до тех пор, пока не началась задача А.

Связь создается перетаскиванием мышью одного отрезка диаграммы Ганта на другой, при этом тип связи по умолчанию определяется как ОН (Окончание - Начало). Предшествующей считается задача, с которой началось перетаскивание, а последующей та, на которой перетаскивание закончилось (на последующую задачу указывает стрелка в конце связи) (рис. 8.17).

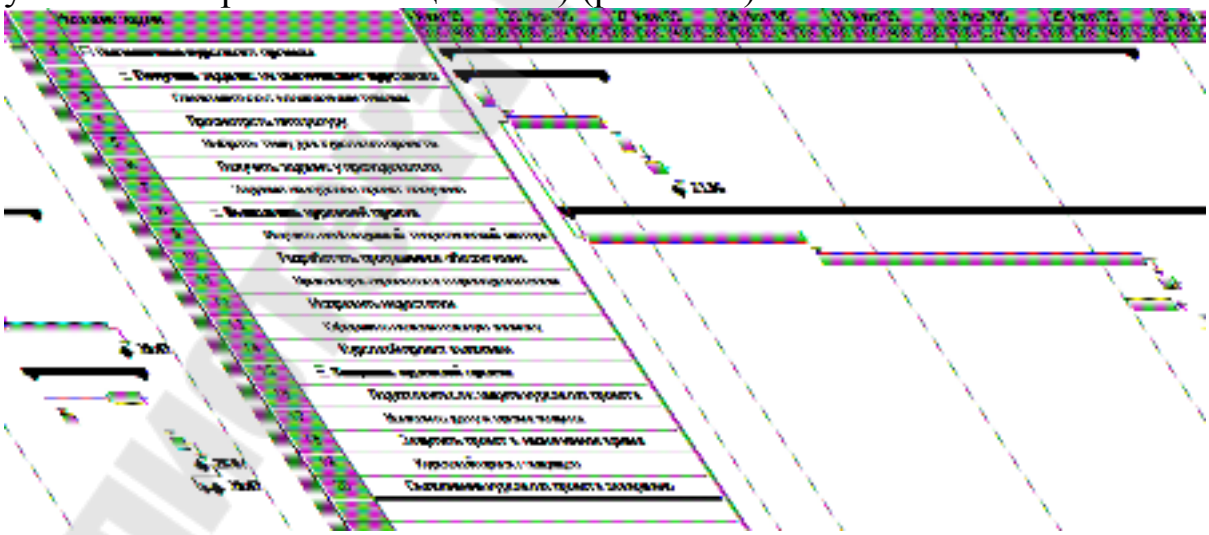


Рис. 8.17. Диаграмма Ганта со связями между задачами

Для удаления связи или изменения ее типа необходимо дважды щелкнуть на ней и произвести соответствующие операции в открывшемся диалоговом окне (рис. 8.18).

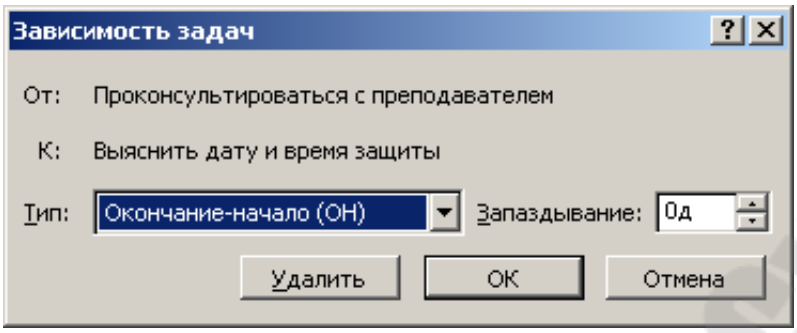


Рис. 8.18. Окно настройки зависимости задач

В раскрывающемся списке *Тип (Type)* можно выбрать тип связи, в поле со счетчиком *Запаздывание (Lag)* указать временной интервал между связанными задачами. Кнопка *Удалить (Delete)* позволяет удалить связь.

8.3 Форматирование диаграмм Ганта. Сетевой график.

Диаграмма Ганта (Gant Chart) названа в честь Генри Ганта (1861-1919). В MS Project диаграмма Ганта является основным средством визуализации плана проекта. Все элементы диаграммы Ганта в MS Project являются настраиваемыми отрезками, каждый из которых может состоять из трех элементов: точки начала, точки окончания и промежуточной части (при этом любой их этих элементов может отсутствовать). При стандартной настройке отрезки, обозначающие фазы, состоят из трех элементов, отрезки, обозначающие задачи – только из промежуточной части, а завершающие задачи только из начальной точки. Длина отрезков, обозначающих фазы и задачи пропорциональна их длительности.

На диаграмме Ганта рядом с отрезками может отображаться дополнительная информация. Состав этой информации определяется настройками программы. В MS Project входит несколько заранее настроенных версий диаграммы Ганта, список которых приведен в табл. 8.1.

Таблица 8.1

Предопределенные версии диаграмм Ганта

Название диаграммы	***Описание***
Подробная диаграмма Ганта (Detail Gantt)	Диаграмма используется при оптимизации плана проекта, когда требуется равномерно распределить нагрузку между ресурсами. На ней отображаются возможные периоды времени, на которые исполнение задачи

Название диаграммы	***Описание***
	можно отложить, не сдвигая срока окончания проекта
Диаграмма Ганта с выравниванием (Leveling Gantt)	Диаграмма используется для выравнивания нагрузки ресурсов. На ней отображаются все изменения, осуществленные в процессе выравнивания
Диаграмма Ганта с отслеживанием (Tracking Gantt)	Диаграмма используется для сравнения запланированных сроков выполнения проекта и реальных сроков исполнения работ. Для каждой задачи и фазы отображаются запланированный и реальный сроки исполнения
Диаграмма Ганта с несколькими планами (Multiple Baseline Gantt)	Диаграмма используется для сравнения трех первых базовых планов проекта
Диаграмма Ганта с ожидаемым планом проекта (PA_Expected Gantt), Диаграмма Ганта с оптимистичным планом проекта (PA_Optimistic Gantt), Диаграмма Ганта с пессимистичным планом проекта (PA_Pessimistic Gantt)	Диаграммы предназначены для анализа плана работ по методу PERT (PERT Analysis, или сокращенно PA). Поскольку метод заключается в построении трех планов (реалистичного, оптимистичного и пессимистичного) и их дальнейшем анализе, то для работы с ним используются три диаграммы

Чтобы воспользоваться предопределенной версией диаграммы Ганта, необходимо выбрать ее название в пункте меню *Вид → Другие представления…*

В том случае, если необходимо изменить вид стандартной диаграммы Ганта или ее версии, можно использовать средства форматирования диаграмм, которые позволяют:

- изменять форму и цвет составляющих диаграмму отрезков;

- определять, какая проектная информация отображается на диаграмме рядом с отрезками;
- отображать дополнительную графическую информацию (например, отклонение от базового плана);
- форматировать шкалу времени, уменьшая или увеличивая масштаб отображения плана.

Чтобы изменить внешний вид отрезков диаграммы Ганта необходимо щелкнуть правой клавишей мыши над необходимым отрезком, в контекстном меню выбрать пункт *Форматировать отрезок…* и изменить необходимые значения настроек.

Существует также возможность группового форматирования элементов диаграммы, которое позволяет полностью настраивать вид диаграммы, определяя, какая именно информация из проектного файла и каким образом отображается на диаграмме. Для этого в диалоговом окне, вызываемом с помощью команды *Формат* → *Стили отрезков* (рисунок 15).

Окно состоит из двух основных блоков. Верхний блок содержит таблицу, в которой определяются отражаемые на диаграмме типы отрезков. Нижний блок содержит две вкладки с параметрами, определяющими внешний вид типов отрезков, указанных в верхнем блоке.

В столбцах таблицы верхнего блока задаются основные свойства типов отрезков диаграмм.

В столбце *Название (Name)* определяется название типа отрезка, которое будет отображаться во всплывающей подсказке при наведении указателя мыши на отрезок.

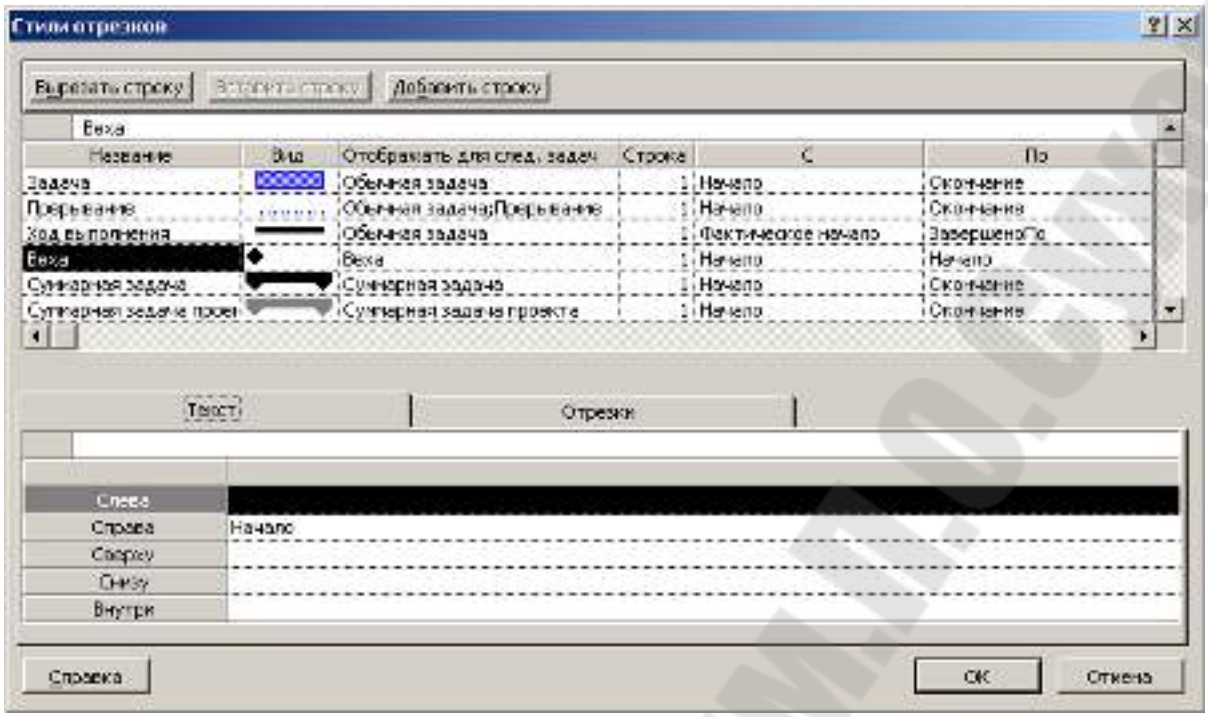


Рис. 8.19. Окно «Стили отрезков»

Поле *Вид (Appearance)* отражает внешний вид элемента диаграммы в соответствии с настройками в нижнем блоке таблицы. Для того, чтобы это поле изменилось, нужно установить курсор в строку с нужным типом отрезков и настроить параметры отображения на вкладках нижней части окна.

Поле *Отображать для след. задач (Show for ...Tasks)* определяет, какие задачи отбирать для отображения данным стилем. Отбор происходит на основании полей типа *Флаг (Flag)*, которые могут содержать только значения *Да (Yes)* и *Нет (No)*. Чтобы определить тип задач, нужно выбрать одно или несколько таких полей в раскрывающемся списке.

В тех случаях, когда необходимо вывести на диаграмме несколько типов фигур для одного типа задач, но не желательно, чтобы они наслаивались друг на друга, можно воспользоваться полем *Строка (Row)*. Номером в поле *Строка* определяется порядок отображения фигур: первым отображается отрезок номер 1, вторым – 2 и т.д.

В полях *С (From)* и *По (To)* определяются начальные и конечные точки отрезка на диаграмме.

Сетевой график

Хотя подавляющее большинство операций по планированию и анализу проекта выполняется с использованием диаграммы Ганта, для работы с сетевым графиком также предусмотрен богатый набор

функций. Наибольший эффект от их применения можно получить на этапе разработки структуры нового проекта «с нуля», а также при анализе проекта с точки зрения возможных рисков.

В отличие от классического сетевого графика, на котором работам соответствуют дуги, а события – вершины, в MS Project используется другой вариант графика, при котором работам соответствуют вершины, а дуги переходам от одной работы к другой.

В MS Project есть три вида графиков, которые можно отнести к сетевым: собственно *сетевой график (Network Diagram)*, *сетевой график с описанием (Descriptive Network Diagram)* и *схема данных (Relationship Diagram)*.

Отличие сетевого графика с описанием от обычного сетевого графика заключается только в повышенной информативности блоков: по умолчанию в них отображается больше информации. Принцип построения и внешний вид схемы данных отличается от первых двух графиков. Разница состоит в том, что на первых двух графиках можно сразу просматривать информацию обо всем плане проекта, а на схеме данных – только об одной задаче этого плана.

Блоки сетевого графика могут различаться цветом и формой в зависимости от типа задачи (обычная задача, завершающая задача или фаза) и ее состояния (исполняется, не исполняется, завершена). На блоке может быть указана любая дополнительная информация: даты начала и окончания, длительность, задействованные ресурсы. По умолчанию фазы обозначаются параллелограммами, задачи – прямоугольниками, завершающие задачи – шестиугольниками. Начатые задачи перечеркиваются одной линией, а завершенные двумя. Стрелки соответствуют связям между задачами.

В MS Project включен большой набор средств для форматирования сетевых графиков. Принципы форматирования сетевого графика не отличаются от принципов форматирования диаграмм Ганта: можно отформатировать отдельный элемент графика, группу элементов и настроить дополнительные параметры группы.

Чтобы отформатировать блок, нужно щелкнуть на нем правой клавишей мыши и выбрать пункт контекстного меню *Формат рамки (Format Box)*.

Возможности форматирования групп блоков сетевого графика завися от того, какие типы задач эти блоки символизируют на графике. Групповое форматирование блоков осуществляется в

диалоговом окне, определяющем свойства групп блоков – пункт меню *Формат – Стили рамок*.

Сетевой график, как и диаграмма Ганта, позволяет редактировать план проекта. Двойной щелчок на блоке позволяет редактировать свойства задачи. Для редактирования отдельного свойства задачи, отображенного на блоке, нужно щелчком мыши установить курсор в соответствующую ячейку блока и затем отредактировать ее.

Для создания задачи (блока) нужно нажать кнопку мыши на свободном месте графика, растянуть на нем прямоугольную рамку, определяющую размер будущего блока, и отпустить кнопку мыши. Создавать задачу на графике не очень удобно, потому что не всегда можно точно предсказать, к какой фазе проекта будет отнесена новая задача. Изменение уровня вложенности задачи производится с помощью комбинаций клавиш Shift+Alt+← и Shift+Alt+→. Удалить задачу можно нажатием клавиши Delete. Связи между задачами создаются перетаскиванием задач друг на друга с помощью мыши, как и в диаграмме Ганта.

Сетевой график удобно настраивать с помощью панели инструментов *Сетевой график (Network Diagram)*, которая вызывается с помощью окна Сервис – Настройка – Панели инструментов.

Схема данных – это особый вид сетевого графика, предназначенный только для анализа связей между задачами проекта. Схема данных разделена на страницы, которые можно пролистывать с помощью полосы прокрутки. На одной странице можно просматривать информацию только об одной задаче. Страницы с задачами расположены в порядке возрастания номеров задач. Диаграмма не позволяет редактировать блоки или проектные данные – их можно только просматривать. Использовать эту диаграмму удобно для анализа плана проекта с множеством зависимостей, поскольку с ее помощью легко сфокусироваться на задачах, связанных с выбранной.

Вопросы для самоконтроля:

1. На какие попросы планирования может помочь ответить модель, созданная в Microsoft Project?
2. Что такое сетевой график?

3. Какие типы связей между задачами существуют в Microsoft Project?

4. Какие виды ресурсов существуют в Microsoft Project?

5. Что такое анализ по методу PERT?

6. Что такое фазы, вехи и завершающие задачи в Microsoft Project?

7. Что такое назначения в Microsoft Project и какие задачи планирования они решают?

ЛИТЕРАТУРА

1. Шафер Д., Фатрел Р., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Вильямс, 2003. – 1136 с.

2. Хелдман К. Профессиональное управление проектом – М.: БИНОМ. Лаборатория знаний, 2005. – 517 с.

3. Ройс У. Управление проектами по созданий программного обеспечения. – М.: Издательство «Лори», 2000. – 431 с.

4. Брауде Э. Технологии разработки программного обеспечения. – СПб.: Питер, 2004. – 655 с.

5. Беркун С. Искусство управления IT-проектами. – СПб.: Питер, 2007. – 400 с.

6. Microsoft Office Project 2007. Библия пользователя. М.: Вильямс, 2008. – 800 с.

7. Microsoft Office Project Professional 2007. Управление проектами: Практическое пособие. – СПб.: Корона-Век, 2008. – 480 с.

8. Богданов В. Управление проектами в Microsoft Office Project 2007. Учебный курс. – СПб.: Питер, 2008. – 592 с.

9. Милошевич Д. Набор инструментов для управления проектами. – М.: Компания АиТи; ДМК Пресс, 2008. – 729 с.

Родионов Андрей Александрович
Андреева Дина Петровна

УПРАВЛЕНИЕ ВЕБ-ПРОЕКТАМИ

Пособие
по одноименной дисциплине
для слушателей специальности 1-40 01 74
«Web-дизайн и компьютерная графика»
заочной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 27.09.12.

Рег. № 26Е.
http://www.gstu.by